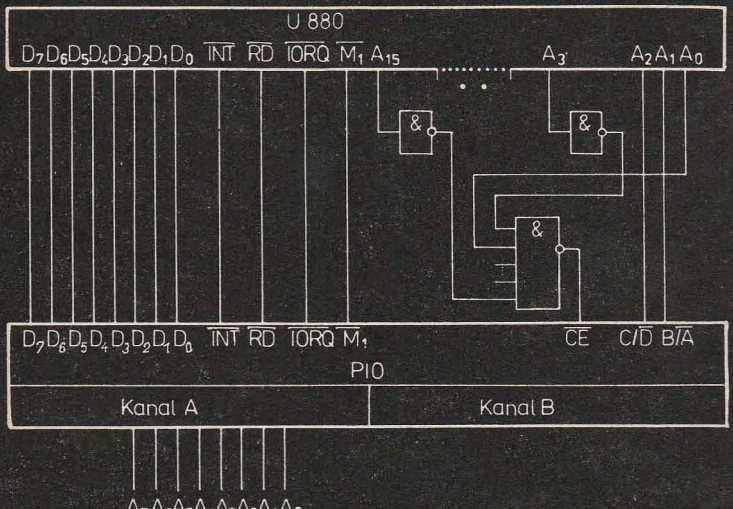
amateurreihe

electronica



Barthold/Bäurich

204

Mikroprozessoren — Mikroelektronische Schaltkreise und ihre Anwendung (Teil 3)

electronica · Band 204

HANS BARTHOLD
DR. HEINZ BÄURICH

Mikroprozessoren – Mikroelektronische Schaltkreise und ihre Anwendung

Teil 3: Periphere Rechnerschaltkreise

MILITÄRVERLAG
DER DEUTSCHEN DEMOKRATISCHEN
REPUBLIK

2. Auflage, 1982
© Militärverlag der
Deutschen Demokratischen Republik (VEB) — Berlin, 1981
Lizenz-Nr. 5 · LSV 3539
Lektor: Wolfgang Stammler
Zeichnungen: Johanna Goernemann
Typografie: Helmut Herrmann
Printed in the German Democratic Republic
Gesamtherstellung: Druckerei Märkische Volksstimme Potsdam
Redaktionsschluß: 15. März 1982
Bestellnummer: 746 418 7
DDR 1,90 M

Inhaltsverzeichnis

1.	**Übersicht über verwendete Abkürzungen**	5
2.	**Ein-/Ausgabe-Bausteine zum Prozessor *U880***	6
2.1.	Grundfunktion der E/A-Bausteine	6
2.2.	Paralleler Ein-/Ausgabe-Baustein *U 880* — PIO	8
2.2.1.	Struktur des *U 880* — PIO	8
2.2.2.	Beschreibung der Anschlüsse des *U 880* — PIO	9
2.2.3.	Arbeitsweise des *U 880* — PIO	11
2.2.4.	Beispiel zum *U 880* — PIO	16
2.3.	Zeitgeberbaustein *U 880* — CTC	18
2.3.1.	Struktur des CTC-Bausteins	18
2.3.2.	Beschreibung der Anschlüsse des CTC-Bausteins	19
2.3.3.	Arbeitsweise des CTC-Bausteins	20
2.4.	Serieller Ein-/Ausgabe-Baustein *U 880* — SIO	22
2.4.1.	Struktur des *U 880* — SIO	22
2.4.2.	Beschreibung der Anschlüsse des *U 880* — SIO	23
2.4.3.	Aufbau eines Kanals des *U 880* — SIO	25
2.4.4.	Arbeitsweise des SIO-Bausteins	27
2.4.5.	Interruptbildung und Statusbildung	30
2.4.6.	Beschreibung der Bit-Stellen der Steuerregister WR_0 bis WR_7 und der Statusregister RR_0 bis RR_2	35
2.5.	DMA-Steuerbaustein *U 880* — DMA	41
2.5.1.	Funktion und Struktur des DMA-Bausteins	41
2.5.2.	Beschreibung der Anschlüsse des DMA-Bausteins	43
2.5.3.	Arbeitsweise des DMA-Bausteins	44
2.5.4.	Beispiel zum DMA-Baustein	50
3.	**Programmier- und Anwendungsbeispiele kleiner Mikrorechner**	53
3.1.	Steuerung des Prozessors	53
3.2.	Bedienelemente	55
3.3.	Anzeige mit LED-Elementen	57
3.4.	Digitaluhr	63
3.4.1.	Digitaluhr mit Prozessor und einer LED-Anzeige	63
3.4.2.	Digitaluhr mit CTC-Baustein	70

3.5. Rechenprogramme 79
3.5.1. Umwandlung einer Dualzahl in eine BCD-Zahl 80
3.5.2. Umwandlung einer BCD-Zahl in eine Dualzahl 83
3.5.3. Programm zur Multiplikation von 2 positiven 16-Bit-Zahlen 86
3.5.4. Division 89
3.5.5. Quadratwurzel 93

4. Schlußbetrachtung 98

Literaturverzeichnis 107

1. Übersicht über verwendete Abkürzungen

Nachstehend sind die verwendeten Formelzeichen und Abkürzungen aufgeführt. Spezielle Zeichen, die nur wenig benutzt werden, sind im Text erläutert.

CE	Chip Enable (Chip-Freigabe)
CLK	Clock (Takt)
CRC	Cyclic Redundancy Check (Zyklische Blockprüfung)
CS	Chip Select (Chip-Auswahl)
CTC	Counter Timer Circuit (Zähler-Zeitgeber-Baustein)
DI	Disable Interrupt (Sperren Interrupt)
DMA	Direct Memory Access (Baustein für direkten Speicherzugriff – Direkter Speicherkanal)
EI	Enable Interrupt (Freigabe Interrupt)
HWT	Höherwertiger Teil von 2 Byte
IEI	Interrupt Enable Input (Interruptfreigabeeingang)
IEO	Interrupt Enable Output (Interruptfreigabe-Weitergabesignal)
INT	Interrupt (Programmunterbrechung)
NWT	Niederwertiger Teil von 2 Byte
PIO	Parallel Input/Output (Paralleler Ein-/Ausgabe-Baustein)
RDY	Ready (Bereitschaft)
RR	Read Register (Statusregister im SIO)
Sgn (p)	Sign (p) = Vorzeichen von p (1, wenn p positiv; −1, wenn p negativ)
SIO	Serial Input/Output (Serieller Ein-/Ausgabe-Baustein)
STB	Strobe (Torungsimpuls)
WR	Write Register (Steuerregister im SIO)

2. Ein-/Ausgabe-Bausteine zum Prozessor *U 880*

2.1. Grundfunktion der E/A-Bausteine

Ein- und Ausgabe-Bausteine bilden das Interface zum Anschluß von peripheren Geräten an den Rechnerbus.
Bild 2.1 zeigt das Prinzip des Anschlusses von einem Bedienpult, eines Lochbandlesers, eines Magnetbandgerätes und eines Lochbandstanzers an den Prozessor *U 880*. Auf die gleiche Weise lassen sich auch AD-Wandler, DA-Wandler, Drucker sowie Bildschirmeinheiten steuern.
Der Ein-/Ausgabe-Baustein erzeugt aus den Signalen des Rechnerbusses die Signale, die zur Steuerung peripherer Geräte erforderlich sind. Das periphere Gerät sendet Fertigmeldung und Fehlermeldungen dem E/A-Baustein, in dem diese Meldungen gesammelt und an den Rechnerbus weitergegeben werden. Die CPU kann solche Meldungen am peripheren Baustein abfragen (Polling). Eine zweite Möglichkeit besteht in der Bildung einer Interruptanfor-

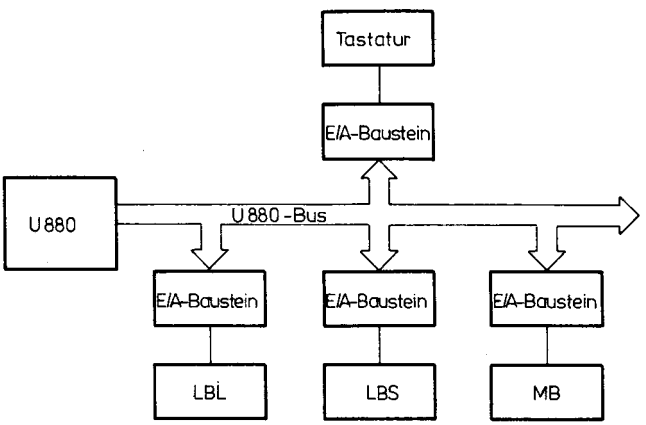


Bild 2.1 Anschluß von Bedienpult, Lochbandleser, Lochbandstanzer und Magnetbandgerät an den Prozessor *U 880*

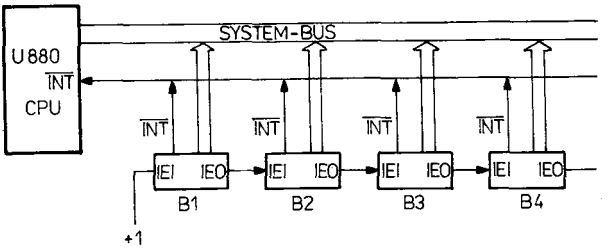


Bild 2.2 Zusammenschaltung von peripheren Bausteinen B1 bis B4 zu einer Interruptprioritätenkette

derung an die CPU. Zur Interruptbildung können die E/A-Bausteine der *U 880*-Familie Interruptketten bilden.

Struktur einer Interruptkette

Jeder E/A-Baustein hat die Anschlüsse IEI, IEO, $\overline{\text{INT}}$. Wenn IEI (interrupt enable input) H-Pegel führt, so kann der Baustein ein Interruptsignal bilden. IEO (interrupt enable output) ist „High", wenn IEI auf H-Pegel liegt und keine Interruptanforderung vom Baustein selbst kommt. Mit den beiden Signalen IEI und IEO lassen sich die Bausteine in einer Kette nach Bild 2.2 zusammenstellen. Der dem Prozessor am nächsten liegende Baustein hat die höchste Priorität. Die folgende Darstellung zeigt den Ablauf eines Interruptvorgangs, wenn erst Baustein B3 und später Baustein B2 einen Interrupt bildet:

	Hintergrundprogramm
	Baustein B3 bildet Interrupt
	IEO vom B3 wird „Low", B4 ist gesperrt, INT-Annahme von B3
Interrupt-	Retten der Register (PUSH Ri)
routine	EI
vom B3	Baustein B2 bildet Interrupt, IEO von B2 wird „Low", B3, B4 sind gesperrt
	INT-Annahme vom B2
Interrupt-	Retten der Register (PUSH Ri)
routine	Füllen der Register (POP Ri)
vom B2	EI
	RETI wird erkannt von B2

Interrupt-	IEO von B2 wird „High“
routine	Füllen der Register (POP Ri)
vom B3	RETI wird erkannt von B3
	IEO von B3 wird „High“
	Hintergrundprogramm rechnet weiter

2.2. Paralleler Ein-/Ausgabe-Baustein *U 880* — PIO

2.2.1. Struktur des *U 880* — PIO

Bild 2.3 zeigt die Grundstruktur des PIO-Bausteins. Der PIO-Baustein hat die Aufgabe, das Rechnerinterface (Rechnerbus) in ein Interface zur Steuerung peripherer Geräte mit paralleler Dateneingabe bzw. -ausgabe umzusetzen. Der Baustein verfügt über eine interne Steuerlogik, die mit Hilfe des Grundtaktes Φ arbeitet, sowie eine Interruptsteuerung mit der Möglichkeit zur Bildung von Prioritätsketten. Die Steuerlogik erzeugt aus den Signalen des Rechnerbusses die zur Peripheriesteuerung notwendigen Signale. Zur Peripheriesteuerung gibt es 2 Ein-/Ausgabe-

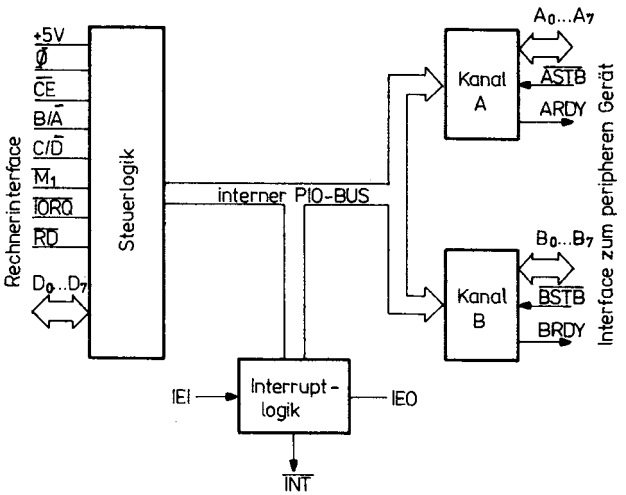


Bild 2.3 Grundstruktur des Bausteins *U 880* — PIO

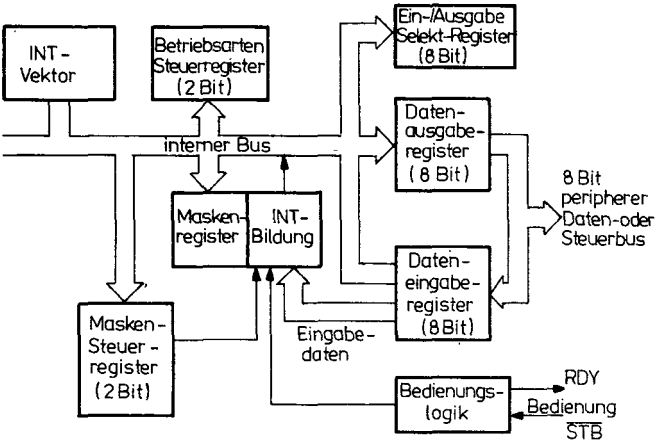


Bild 2.4 Logische Struktur eines Kanals des Bausteins *U 880* — PIO

Kanäle A und B, die im Hand-shake-Verfahren den Transfer zu einem peripheren Gerät realisieren. Jeder Kanal hat

- 1 Eingaberegister (8 Bit),
- 1 Ausgaberegister (8 Bit),
- 1 Selektregister (Auswahlregister; 8 Bit),
- 1 Maskenregister (8 Bit),
- 1 Maskensteuerungsregister (2 Bit),
- 1 Betriebsartenregister (2 Bit).

Bild 2.4 zeigt die logische Struktur eines Kanals. Ein Kanal kann in folgenden Betriebsarten arbeiten:

- byteweise Ausgabe,
- byteweise Eingabe,
- byteweise bidirectional (über dieselbe Leitung können Informationen ein- bzw. ausgegeben werden),
- Einzelbitsteuerung.

2.2.2. Beschreibung der Anschlüsse des *U 880* — PIO

$D_0 \ldots D_7$ Datenbus zum Prozessor *U 880* (bidirectional, tristate)

B/$\overline{A}$ - Kanalauswahl: legt fest, über welchen Kanal der Datentransfer zwischen *U 880* und PIO stattfindet (Eingabesignal): „Low" = Tor A, „High" = Tor B.

C/$\overline{D}$–Auswahl Steuerwort/Datenwort: legt fest, ob das von *U 880* kommende Wort ein Steuerwort oder ein Datenwort ist (Eingabesignal) „Low" = Datenwort, „High" = Steuerwort.

$\overline{CE}$ Chipauswahlsignal (Eingabesignal, aktiv „Low").

$\overline{M1}$ Signal für M1-Zyklus: entspricht dem Signal M1 des Prozessors *U 880*. Es dient zur Synchronisierung der PIO-Logik (Eingabesignal, aktiv „Low").

$\overline{IORQ}$ Ein-/Ausgabe-Anforderungssignal: entspricht dem $\overline{IORQ}$ des Prozessors *U 880*. Es dient der Ansteuerung des Bausteins bei einem Ein- und Ausgabebefehl. Bei gleichzeitigem Erscheinen von IORQ und M1 gibt das Interrupt erzeugende Tor automatisch einen Interruptvektor an den Datenbus ab (Eingabesignal, aktiv „Low").

$\overline{RD}$ Lesesignal: entspricht dem READ-Signal des Prozessors *U 880* (Eingabesignal, aktiv „Low").
Der Weg der Daten oder Befehle bei der Ein- und Ausgabe vom *U 880* zum PIO wird durch die Signale $\overline{IORQ}$, $\overline{RD}$, B/$\overline{A}$, C/$\overline{D}$, $\overline{CE}$ gesteuert.

IEI Interruptfreigabesignal: dient zur Bildung von Interruptprioritätsschaltungen (Eingabesignal, aktiv „High").

IEO Signal für die Weitergabe der Interruptfreigabe, es ist das zu IEI gehörende Ausgangssignal.
Es ist „High", wenn IEI gleich „High" ist und keine Interruptanforderung vom Baustein selbst kommt. Im anderen Fall ist es „Low" (Ausgabesignal).

$\overline{INT}$ Interruptanforderungssignal; wenn das Signal $\overline{INT}$ aktiv ist, fordert das PIO einen Interrupt im Prozessor (Ausgabesignal, aktiv „Low").

$A_0 \ldots A_7$ Datenbuskanal A. $A_0 \ldots A_7$ dient zur Ein- und Ausgabe der Information zwischen Kanal A und einem peripheren Gerät. A_0 ist die niederwertigste Stelle.

$\overline{ASTB}$ Strobeimpuls (Torungsimpuls) für Kanal A (Eingabesignal, aktiv „Low").

ARDY Bereitschaftssignal für Tor A: Bei Ausgabe sagt dieses Signal aus, daß das Ausgaberegister Daten enthält. Bei Eingabe bedeutet dieses Signal, daß der Inhalt des Eingaberegisters abgeholt worden ist und der Kanal bereit ist, neue Daten zu empfangen (Ausgabesigna , aktiv „High").

$B_0 \ldots B_7$ Datenbuskanal B. $B_0 \ldots B_7$ dient zur Ein- und Ausgabe der Information zwischen Kanal B und einem peripheren Gerät. B_0 ist die niederwertigste Stelle.

$\overline{\text{BSTB}}$ Strobeimpuls (Torungsimpuls) für Kanal B (Eingabesignal, aktiv „Low").

BRDY Bereitschaftssignal für Tor B: Bei Ausgabe sagt dieses Signal aus, daß das Ausgaberegister Daten enthält. Bei Eingabe bedeutet dieses Signal, daß der Inhalt des Eingaberegisters abgeholt worden ist und der Kanal bereit ist, neue Daten zu empfangen (Ausgabesignal, aktiv „High").

Ein spezielles Löschsignal (RESET) gibt es nicht. Das Löschen der Register und der Flip-Flop des Bausteins geschieht entweder beim Anlegen der Betriebsspannung oder durch $\overline{\text{MI}}$, wenn nicht gleichzeitig eines der Signale $\overline{\text{RD}}$ bzw. $\overline{\text{IORQ}}$ aktiv ist. Dieses Löschen erzeugt folgenden Zustand:

- die Maskenregister sind gelöscht;
- beide Interrupt-Erlaube-Flip-Flop sind gelöscht;
- beide Ausgaberegister sind gelöscht;
- das Register zur Bereitstellung des Interruptvektors wird nicht gelöscht;
- Tor A und B werden hochohmig, die Hand-shake-Signale $\overline{\text{ASTB}}$, ARDY, $\overline{\text{BSTB}}$, BRDY werden inaktiv.

Der RESET-Zustand bleibt so lange bestehen, bis das PIO ein Steuerwort vom Prozessor erhält.

2.2.3. Arbeitsweise des *U 880* — PIO

Bevor eine Ein- bzw. Ausgabe zum peripheren Gerät vorgenommen werden kann, müssen der Arbeitsmodus und die Interruptlogik eingestellt sein. Dazu werden eine Reihe Steuerwörter in das PIO gespeichert. (Das PIO wird initialisiert.)

Zuerst muß vom Prozessor ein Steuerwort zur Auswahl des Arbeitsmodus gegeben werden. Das Steuerwort hat folgenden Aufbau:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
M	M	x	x	1	1	1	1

Die Stellen D_4 und D_5 sind ohne Bedeutung. Durch die Stellen D_6 und D_7 wird der Arbeitsmodus festgelegt.

	D_7	D_6	Arbeitsmodus
Modus 0	0	0	Ausgabe
Modus 1	0	1	Eingabe
Modus 2	1	0	bidirectional
Modus 3	1	1	Einzelbitsteuerung

Modus 0 Ausgabe:
Nach dem Übergeben des Steuerwortes können bei der Arbeit mit Interrupt ein Interruptvektor und ein Interruptsteuerwort folgen. Der *Interruptvektor* hat folgenden Aufbau:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	V_2	V_1	0

Das *Interruptsteuerwort* hat folgenden Aufbau:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
M	x	x	x	0	0	1	1

Ist D_7 gleich 1, so wird die Interruptbildung erlaubt (Bildung des Signals INT). Ist D_7 gleich 0, dann wird die Interruptbildung gesperrt. Nach Speicherung von Interruptsteuerwort und Interruptvektor ist der PIO-Baustein programmiert bzw. initialisiert. Es können die eigentlichen Ausgabebefehle zur Datenausgabe folgen.
Durch den Ausgabebefehl vom Prozessor wird das Ausgaberegister gefüllt und das Signal RDY (entweder ARDY oder BRDY) aktiv. Dieses Signal bleibt so lange aktiv, bis von einem peripheren Gerät das Signal $\overline{\text{STB}}$ ($\overline{\text{ASTB}}$, $\overline{\text{BSTB}}$) kommt. Mit dem Signal

$\overline{STB}$ werden gewöhnlich die Daten vom Ausgaberegister in das periphere Gerät übernommen. Ist die Bildung einer Interruptanforderung erlaubt, dann wird durch $\overline{STB}$ das Signal $\overline{INT}$ erzeugt. Das Signal $\overline{INT}$ führt zu einer Programmunterbrechung im Prozessor. Bei Annahme des Interrupt werden gleichzeitig die Signale $\overline{M1}$ und $\overline{IORQ}$ aktiv. Als Folge der Signale $\overline{M1}$ und $\overline{IORQ}$ gibt das PIO den Interruptvektor auf den Datenbus. Mit Hilfe des Interruptvektors bestimmt der Prozessor die Startadresse des Interruptbedienprogramms.

Modus 1 Eingabe:

Nach dem Übergeben des Steuerwortes können bei der Arbeit mit Interrupt ein Interruptvektor und ein Interruptsteuerwort folgen. Der *Interruptvektor* hat folgenden Aufbau:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	V_2	V_1	0

Das *Interruptsteuerwort* hat folgenden Aufbau:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
M	x	x	x	0	0	1	1

Ist D_7 gleich 1, so wird die Interruptbildung erlaubt (Bildung des Signals $\overline{INT}$). Ist D_7 gleich 0, dann wird die Interruptbildung gesperrt. Nach Speicherung von Interruptvektor und Interruptsteuerwort können Eingabebefehle zur Dateneingabe folgen.

Nach einem Eingabebefehl vom Prozessor ist das Eingaberegister leer und das Signal RDY (ARDY, BRDY) aktiv. Dieses Signal bleibt so lange aktiv, bis von einem peripheren Gerät das Signal $\overline{STB}$ ($\overline{ASTB}$ oder $\overline{BSTB}$) kommt. Mit der Vorderflanke des Signals $\overline{STB}$ können die Daten vom peripheren Gerät in das Eingaberegister übernommen werden.

Ist die Bildung einer Interruptanforderung erlaubt, so wird durch $\overline{STB}$ das Signal $\overline{INT}$ erzeugt. Das Signal $\overline{INT}$ führt zu einer Programmunterbrechung im Prozessor. Bei Annahme des Interrupt werden gleichzeitig die Signale $\overline{M1}$ und $\overline{IORQ}$ aktiv. Als Folge der Signale $\overline{M1}$ und $\overline{IORQ}$ gibt das PIO den Interruptvektor auf den Datenbus. Mit Hilfe des Interruptvektors bestimmt der Prozessor die Startadresse des Interruptbedienprogramms.

Modus 2 bidirectional:
Eine bidirectionale Ein- und Ausgabe ist nur über Kanal A möglich. Dabei werden die Hand-shake-Signale von Kanal A (ARDY, $\overline{\text{ASTB}}$) zur Ausgabe und die Signale von Kanal B (BRDY, $\overline{\text{BSTB}}$) zur Eingabe benutzt. Vor der Benutzung von Kanal A als bidirectionaler Datenbus muß Kanal B in den Modus Einzelbitsteuerung gebracht werden.
Nach dem Übergeben des Steuerworts können bei der Arbeit mit Interrupt ein Interruptvektor und ein Interruptsteuerwort folgen. Der *Interruptvektor* hat folgenden Aufbau:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	V_2	V_1	0

Das *Interruptsteuerwort* hat wieder folgenden Aufbau:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
M	x	x	x	0	0	1	1

Ist D_7 gleich 1, so wird die Interruptbildung erlaubt (Bildung des Signals $\overline{\text{INT}}$). Ist D_7 gleich 0, dann wird die Interruptbildung gesperrt.

Nach der Übergabe von Interruptsteuerwort und Interruptvektor können die Ein- und Ausgabebefehle zum Datentransport gegeben werden. Nach einem Ausgabebefehl vom Prozessor ist das Ausgaberegister gefüllt und das Signal ARDY zur Kennzeichnung der Bereitschaft aktiv.
Eine Abnahme der Daten vom Ausgaberegister kann während der Zeit geschehen, in der $\overline{\text{ASTB}}$ und ARDY aktiv sind. Mit der Rückflanke von $\overline{\text{ASTB}}$ wird bei erlaubtem Interrupt das INT-Signal gebildet. Nach einem Eingabebefehl ist das Signal BRDY aktiv.

Mit dem Signal $\overline{\text{BSTB}}$ können neue Daten von einem peripheren Gerät in das Eingaberegister übernommen werden. Anschließend wird das Signal BRDY inaktiv. Auch in diesem Fall bietet man mit der Rückflanke von $\overline{\text{BSTB}}$ das Interruptsignal bei erlaubtem Interrupt.
Modus 3 Einzelbitsteuerung:
Bei der Einzelbitsteuerung werden die Datenleitungen der Kanäle A und B als einzelne Steuerleitungen verwendet. Die Hand-

shake-Signale haben in diesem Arbeitsmodus keine Bedeutung. Der Modus dient zur Aufnahme von Alarm- und Statussignalen sowie zur Ausgabe von Steuersignalen. Nach der Ausgabe des *Steuerwortes* mit dem Aufbau

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	x	x	1	1	1	1

muß ein weiteres Byte folgen, das festlegt, welche Datenleitungen des Kanals Ein- bzw. Ausgangsleitung sein sollen (Auswahlbyte). Ist eine Bit-Stelle dieses Bytes 0, so ist die entsprechende Datenleitung des Kanals Ausgangsleitung. Bei einer 1 in der Bit-Stelle ist die dazugehörige Datenleitung des Kanals eine Eingangsleitung. Wenn z. B. dieses Auswahlbyte den Aufbau

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	1	0	1	0	0	1

hat, so sind die Anschlüsse 0, 3 und 5 Eingangsleitungen und die Anschlüsse 1, 2, 4, 6 und 7 Ausgangsleitungen.
An das Ein-/Ausgabe-Auswahlbyte kann sich bei der Arbeit mit Interrupt ein Interruptvektor mit dem Aufbau

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	V_2	V_1	0

anschließen. Nach dem Interruptvektor folgt das Interruptsteuerwort. Es hat den Aufbau:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Interrupt erlaubt	ODER UND	Tief Hoch	Maske folgt	0	1	1	1

- Ist $D_7 = 1$, so ist die Interruptbildung erlaubt.
- Bei $D_7 = 0$ ist die Interruptbildung gesperrt.
- Ist $D_6 = 1$, so entsteht im PIO das Signal $\overline{\text{INT}}$, wenn alle als Eingang festgelegten Leitungen das durch D_5 definierte Potential haben (UND-Funktion).
- Bei $D_6 = 0$ wird im PIO das Signal $\overline{\text{INT}}$ gebildet, wenn eine der als Eingang gekennzeichneten Leitungen das durch D_5 definierte Potential hat (ODER-Funktion).

- Ist $D_5 = 1$, so führt ein „Hochpegel" an den Eingangsleitungen zum Interrupt.
- Bei $D_5 = 0$ führt ein „Tiefpegel" an den Eingangsleitungen zum Interrupt.
- Ist $D_4 = 1$, so muß dem Steuerwort eine Interruptmaske folgen.

  Interruptmaske

M_7	M_6	M_5	M_4	M_3	M_2	M_1	M_0

  Zur Interruptbildung werden in diesem Fall nur die Eingänge zugelassen, deren Bit-Stellen in der Interruptmaske 0 sind.
- Bei $D_4 = 0$ braucht keine Interruptmaske zu folgen.

2.2.4. Beispiel zum *U 880* — PIO

An einem PIO Kanal A sollen 3 Steuerleitungen (Ausgabe) und 5 Sensorleitungen (Eingabe) angeschlossen werden.
Dazu werden die Leitungen A_0, A_1, A_2 als Ausgangsleitungen, die Leitungen A_3 bis A_7 als Eingangsleitungen festgelegt. Das Ein-/Ausgabe-Auswahlbyte muß folgenden Aufbau haben:

Ein-/Ausgabe-Auswahlbyte:

1	1	1	1	1	0	0	0

Im Prozessor soll ein Interrupt entstehen, wenn die Sensorsignale A_4, A_5 und A_7 gleichzeitig „Hochpegel" aufweisen. Für diesen Fall sieht das Interruptsteuerwort folgendermaßen aus:

Interruptsteuerwort:

1	1	1	1	0	1	1	1

Interruptmaske:

0	1	0	0	1	1	1	1

Das PIO sei nach Bild 2.5 mit dem Prozessor *U 880* verbunden.
- Wenn das Adreß-Bit $A_0 = 1$ und die Adreß-Bits A_3 bis A_{15} gleich 0 sind, wird dieses PIO angesteuert.
- Bei $A_1 = 0$ wird von diesem PIO Kanal A ausgewählt.
- Wenn $A_2 = 1$ ist, wird über den Datenbus ein Steuerwort, wenn $A_2 = 0$ ist, ein Datenwort transportiert.

Für die Steuer- und Datenworte nach Kanal A und B ergeben sich aus dieser Festlegung folgende Adressen:

Steuerwort nach Kanal A	Adresse 5
Datenwort von und nach Kanal A	Adresse 1
Steuerwort nach Kanal B	Adresse 7
Datenwort von und nach Kanal B	Adresse 3

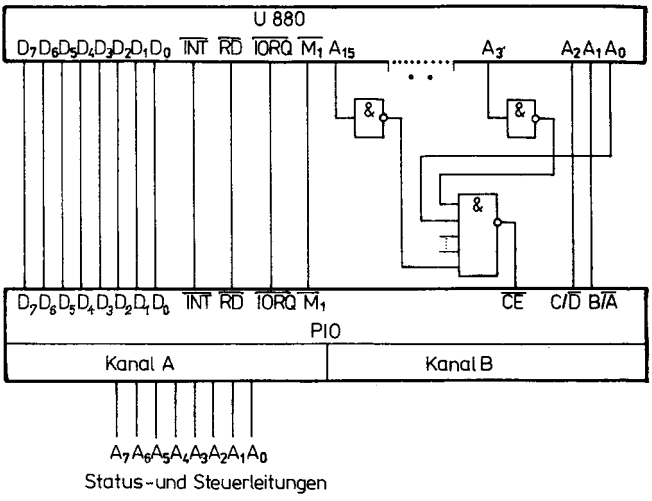


Bild 2.5 Zusammenschaltung eines PIO-Schaltkreises mit dem *U 880*

Für die Initialisierung des geforderten Zustands sind folgende Befehle notwendig:

	Befehl	Wort auf Datenleitung
Ausgabe des Steuerworts zur Festlegung von Bit-Mode Eingabe/Ausgabe	OUT 5	1 1 0 0 1 1 1 1
Auswahlbyte	OUT 5	1 1 1 1 1 0 0 0
Ausgabe Interruptvektor	OUT 5	0 0 0 0 1 0 0 0
Ausgabe Interruptsteuerwort	OUT 5	1 1 1 1 0 1 1 1
Ausgabe Interruptmaske	OUT 5	0 1 0 0 1 1 1 1

Nach dieser Befehlsfolge wird im Prozessor ein Interrupt gefordert, wenn die Leitungen A_4, A_5, A_7 gleichzeitig „Hochpegel" aufweisen. Durch den Eingabebefehl IN 1 kann festgestellt werden, welche Sensorleitungen Hoch- bzw. Tiefpegel haben. Das durch den Befehl IN 1 in den Prozessor *U 880* eingelesene Datenbyte setzt sich zusammen aus den Bit-Stellen 3 bis 7 des Eingaberegisters und den Bit-Stellen 0 bis 2 des Ausgaberegisters von Kanal A.

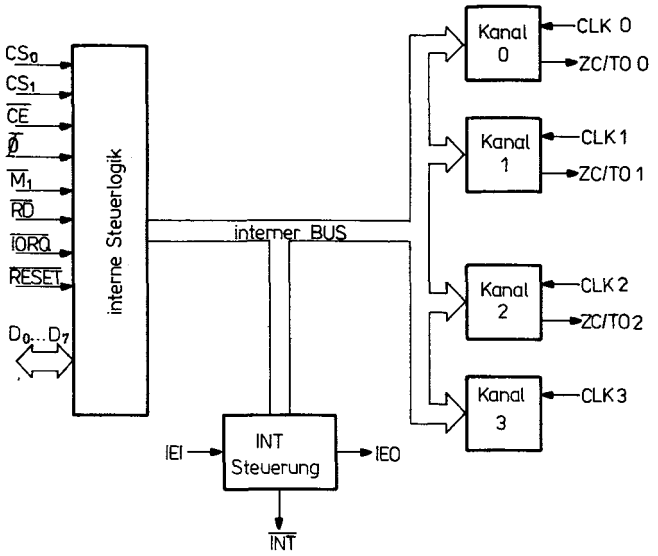


Bild 2.6 Grundstruktur des Bausteins *U 880* — CTC

Weitere Bausteine für die parallele Eingabe sind die Schaltkreise *8255* und *8212*. Der Baustein *8255* hat einen dem PIO ähnlichen Aufbau und läßt sich in gleicher Weise einsetzen. Der Schaltkreis *8212* ist in Teil 1, Abschnitt 3.4.2., beschrieben.

2.3. Zeitgeberbaustein *U 880* — CTC

2.3.1. Struktur des CTC-Bausteins

Bild 2.6 zeigt die logische Struktur des CTC (Counter-Timer-Circuit – Zähler-Zeitgeber-Schaltkreis). Der Baustein CTC ist ein peripherer Baustein zum Prozessor *U 880*. Er dient sowohl als Zähl- wie auch als Zeitgeberbaustein. Mit ihm lassen sich auf 4 getrennten Kanälen Zähler- sowie Zeitgeberfunktionen realisieren. Rechnerseitig hat der Baustein das zum *U 880* passende Interface mit dem 8-Bit-Datenbus und den Steuersignalen $\overline{\text{IORQ}}$, $\overline{\text{RD}}$, $\overline{\text{M}_1}$,

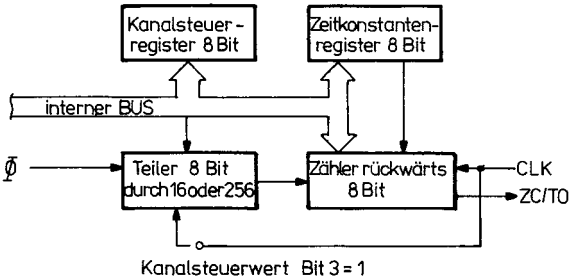


Bild 2.7 Logische Struktur eines Kanals des Bausteins *U 880* — CTC

Systemtakt Φ, $\overline{\text{RESET}}$, $\overline{\text{CE}}$, CS_1, CS_0. Zur Realisierung von Interruptketten verfügt der Baustein über die Signale IEI, IEO, $\overline{\text{INT}}$.

Bild 2.7 zeigt den Aufbau eines einzelnen Kanals. Hauptbestandteil des Kanals ist ein 8-Bit-Rückwärtszähler, der über das Zeitkonstantenregister eingestellt werden kann. Die Zählimpulse können über den Takteingang CLK oder über einen Vorteiler kommen. Der Ausgang ZC/TO zeigt den Nulldurchgang des Zählers nach außen. Zur Einstellung der Betriebsarten hat jeder Kanal ein Kanalsteuerregister.

2.3.2. Beschreibung der Anschlüsse des CTC-Bausteins

Im einzelnen haben die Anschlüsse folgende Bedeutung:

$D_0 \ldots D_7$ Datenbus zum Prozessor (bidirectional).

$\overline{M_1}$, $\overline{\text{IORQ}}$, $\overline{\text{RD}}$ Steuersignale vom Prozessor (Eingang, aktiv „Low").

$\overline{\text{CE}}$ Chipauswahlsignal (Eingang, aktiv „Low").

CS_0, CS_1 Enthält die Nummer des auszuwählenden Kanals (Eingang, aktiv „High").

CS_1	CS_0	Kanal-Nr.
0	0	0
0	1	1
1	0	2
1	1	3

$\overline{\text{INT}}$	Interruptanforderung vom CTC an den Prozessor (Ausgang, aktiv „Low“).
IEI	Interruptfreigabesignal (Eingabe, aktiv „High“). Ist IEI aktiv („High“), dann kann der Baustein eine Interruptanforderung bilden.
IEO	Interruptfreigabesignal für den nächsten in der Kette liegenden Ein-/Ausgabe-Baustein (Ausgabe, aktiv „High“). Der Ausgang IEO hat nur dann Hochpotential, wenn im Baustein keine Interruptanforderung gebildet wird und IEI ebenfalls Hochpotential hat. Das heißt, existiert im Baustein und links vom Baustein keine Interruptanforderung, dann werden die Bausteine auf der rechten Seite der Kette freigegeben.
CLK 0–3	Externer Zähleingang für die Kanäle 0 bis 3 (Eingang, aktiv „High“).
ZC/TO 0–2	Zeigt den Nulldurchgang der Zähler der Kanäle 0 bis 2 an (Ausgang, aktiv „High“).
$\overline{\text{RESET}}$	Rücksetzsignal. Dieses Signal stellt die Zähler auf 0, setzt die Interruptfreigabe-Bits zurück (Eingang, aktiv „Low“).

2.3.3. Arbeitsweise des CTC-Bausteins

Jeder Kanal kann in der Betriebsart Zähler oder Zeitgeber arbeiten. In der *Betriebsart Zähler* wird mit der steigenden Flanke des Eingangs CLK der Zähler um 1 rückgezählt. Der Zähler läuft synchron mit dem Systemtakt Φ, d. h., das Rückwärtszählen nach der steigenden Flanke von CLK erfolgt mit dem nächsten Takt. Bei Nulldurchgang des Zählers wird eine Interruptanforderung gebildet und die Zeitkonstante in den Zähler geladen. In der *Betriebsart Zeitgeber* wird der Rückwärtszähler über einen Vorteiler mit dem Systemtakt Φ getaktet. Der Vorteiler läßt sich vom externen Eingang CLK triggern. Der Triggerzeitpunkt wird durch Bit D_3 im Kanalsteuerwort bestimmt. Es gilt:

D_3	D_2 (Kanalsteuerwort)
0	0 Zeitmessung beginnt am Anfang des nächsten Maschinenzyklus.

0 1 Zeitmessung beginnt am Anfang des nächsten Maschinenzyklus nach dem Laden der Zeitkonstante.

1 0 Zeitmessung beginnt am Anfang des nächsten Maschinenzyklus, nach dem der Eingang CLK die vorgesehene Flanke durchläuft.

1 1 Zeitmessung beginnt am Anfang des nächsten Maschinenzyklus, nach dem die Zeitkonstante geladen ist und die vorgesehene Flanke CLK am Eingang erscheint.

Beim Nulldurchgang des Zählers wird ein Interrupt gebildet, der zur Realisierung eines Uhrenprogramms mit Hilfe des Mikroprozessors verwendet werden kann. Die Interruptperiode t_i errechnet sich nach

$t_i = t_c \cdot P \cdot TK;$

mit t_c — Systemtaktperiode, P — Vorteiler (16 oder 256),
TK — Zeitkonstante.

Programmierung des CTC (Initialisierung)

a) Einschreiben des Interruptvektors
Der Interruptvektor hat folgendes Format:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	x	x	0

Die Bit-Stellen D_1 und D_2 werden im Baustein bei Interruptanforderung mit der betreffenden Kanalnummer belegt.

b) Einschreiben des Kanalsteuerworts
Das Kanalsteuerwort hat folgenden Aufbau:

D_7
INT Freigabe, 0 gesperrt, 1 frei
D_6
Betriebsart, 0 Zeitgeber, 1 Zähler
D_5
0 Teiler 16, 1 Teiler 256
D_4
0 neg. Flanke, 1 pos. Flanke
D_3
Bestimmt Triggerzeitpunkt
D_2
Zeitkonstante laden, 0 kein Laden, 1 das nächste Steuerwort für den angewählten Kanal wird als Zeitkonstante interpretiert.

(nur in der Betriebsart „Zeitgeber")

Wird eine Zeitkonstante während eines Zeitmeßvorgangs eingegeben, so erfolgt die Übernahme der Zeitkonstanten in den Zähler erst, wenn der Meßvorgang beendet ist.

D_1

Rücksetzen; 0 Kanalzähler zählt weiter, 1 Kanal unterbricht das Zählen, bis eine Zeitkonstante eingegeben wird.

ZC/TO = inaktiv; Interrupt ist gesperrt.

D_0

1 Kanalsteuerwort, 0 Interruptvektor.

c) Laden der Zeitkonstante
Ist im Kanalsteuerwort das Laden einer Zeitkonstanten gefordert, so wird die Zeitkonstante nach dem Kanalsteuerwort eingeschrieben. Die Zeitkonstante ist ein Dualwert zwischen 0 und 255.

d) Lesen des Rückwärtszählers
Der momentane Wert des Rückwärtszählers kann zu jedem beliebigen Zeitpunkt vom Prozessor aus, mit Hilfe eines Eingabebefehls, gelesen werden. Der Wert repräsentiert die Anzahl der positiven Flanken vor der positiven Flanke von T2 des Lesezyklus.

Interruptbildung

Liegt im Baustein eine Bedingung zur Bildung einer Interruptanforderung vor (Nulldurchgang eines Zählers), so wird der $\overline{\text{INT}}$-Ausgang aktiv, wenn die Interruptbildung erlaubt ist.

2.4. Serieller Ein-/Ausgabe-Baustein *U 880* — SIO

2.4.1. Struktur des *U 880* — SIO

Bild 2.8 zeigt die Grundstruktur des SIO-Bausteins (Seriell Input/Output). Der SIO-Baustein setzt das Rechnerinterface (Rechnerbus) in ein Interface zur Steuerung peripherer Geräte mit serieller Dateneingabe bzw. -ausgabe um. Der Baustein hat eine interne Steuerlogik, die mit Hilfe des Grundtakts Φ arbeitet, sowie eine Interruptsteuerung mit der Möglichkeit zur Bildung von Prioritätsketten. Zur Peripheriesteuerung gibt es 2 Kanäle. Über jeden Kanal können Informationen bitseriell gesendet und empfangen werden.

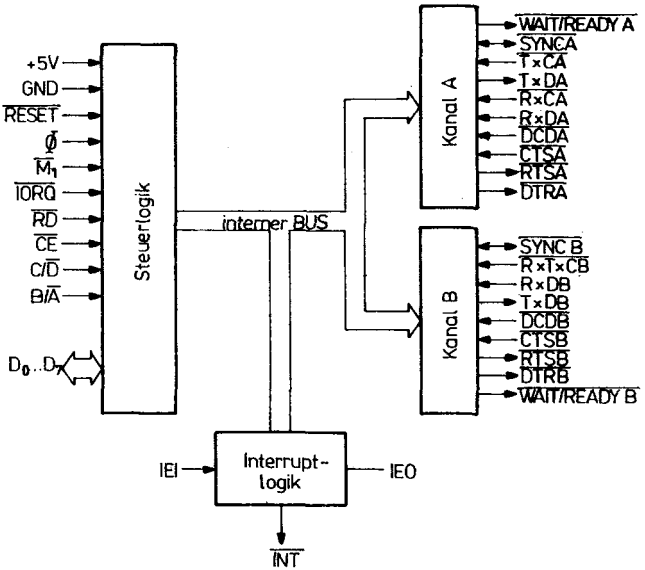


Bild 2.8 Grundstruktur des Bausteins *U 880* — SIO

2.4.2. Beschreibung der Anschlüsse des *U 880* — SIO

Signale des Rechnerinterface:

$D_0 \ldots D_7$	Datenbus zum Prozessor *U 880*.
B/$\overline{\text{A}}$ Eingangssignal	Kanalauswahl; „Low" ≙ Tor A; „High" ≙ Tor B.
C/$\overline{\text{D}}$ Eingangssignal	Auswahl Steuerwort/Datenwort: Dieses Signal legt fest, ob das vom Prozessor kommende Wort ein Steuerwort "(High") oder ein Datenwort („Low") ist.
$\overline{\text{CE}}$ Eingangssignal	Chipauswahlsignal (aktiv ≙ „Low").
$\overline{\text{M1}}$ Eingangssignal	Signal für M1-Zyklus: Dieses Signal entspricht dem Signal $\overline{\text{M1}}$ des Prozessors.

$\overline{RD}$ Eingangssignal	Lesesignal: Dieses Signal entspricht dem $\overline{RD}$-Signal des Prozessors (aktiv $\triangleq$ „Low“).
$\overline{IORQ}$ Eingangssignal	Eingabe-/Ausgabe-Anforderung vom Prozessor (aktiv = „Low“).
IEI Eingangssignal	Interruptfreigabesignal: Dieses Signal dient zur Bildung von Interruptprioritätsschaltungen.
IEO Ausgangssignal	Signal für die Weitergabe der Interruptfreigabe: Bei diesem Signal handelt es sich um das Ausgangssignal zu IEI. Es ist „High“, wenn IEI „High“ ist und keine Interruptanforderung vom SIO-Baustein kommt. Sonst ist es „Low“.
$\overline{INT}$ Ausgangssignal	Interruptanforderungssignal (aktiv $\triangleq$ „Low“).
Φ Eingangssignal	Systemtakt.
$\overline{RESET}$	Rücksetzen der Empfänger- und Senderlogik. Auf T × DA und T × DB entsteht das Grundsignal. Steuerregister sind gelöscht. Alle Interrupts sind gesperrt (aktiv $\triangleq$ „Low“).
Signale der Kanäle A und B:	
R × DA, R × DB Eingangssignale	Empfangsdaten (aktiv $\triangleq$ „High“).
T × DA, T × DB Ausgangssignale	Sendedaten (aktiv $\triangleq$ „High“).
$\overline{R \times CA}$, $\overline{R \times CB}$*) Eingangssignale	Empfängertakt (aktiv $\triangleq$ „Low“).
$\overline{T \times CA}$, $\overline{T \times CB}$*) Eingangssignale	Sendertakt (aktiv $\triangleq$ „Low“).
$\overline{CTSA}$, $\overline{CTSB}$ Eingangssignale	Steuersignal für Datensender. Im Arbeitsmodus AUTOENABLE werden bei $\overline{CTS}$ = „High“ Daten nicht gesendet. In anderen Arbeitsmode können diese Anschlüsse als Eingänge anderweitig verwendet werden. Die beiden Leitungen haben *Schmitt*-Trigger-Eingänge (aktiv $\triangleq$ „Low“).
$\overline{DCDA}$, $\overline{DCDB}$	Steuersignal für Datenempfang. Das Signal

Eingangssignale	$\overline{DCD}$ arbeitet analog dem Signal $\overline{CTS}$, jedoch wird durch $\overline{DCD}$ der Empfänger des zugehörigen Kanals gesperrt (aktiv $\triangleq$ „Low").
$\overline{RTSA}$, $\overline{RTSB}$ Ausgangssignale	Programmierbares Quittungssignal. Sobald das RTS-Bit eines Kanals gesetzt ist, geht der zugehörige $\overline{RTS}$-Ausgang in den „Low"-Zustand. Wird das RTS-Bit im Asynchronmodus rückgesetzt, geht der zugehörige $\overline{RTS}$-Ausgang in den „High"-Zustand, sobald das Senderegister leer ist. Im Synchronmodus folgt der Anschluß dem RTS-Bit (aktiv $\triangleq$ „Low").
$\overline{DTRA}$, $\overline{DTRB}$*) Ausgangssignale	Programmierbares Quittungssignal. Der Ausgang folgt dem Wert des DTR-Bit (aktiv $\triangleq$ „Low").
$\overline{SYNCA}$, $\overline{SYNCB}$	Synchronisationssignal (aktiv $\triangleq$ „Low"). Bei interner Synchronisation: „Ausgang". Das Signal ist in dem Takt aktiv, in dem ein Synchronisationszeichen erkannt wird. Bei externer Synchronisation: „Eingang". Der Aufbau eines Zeichens beginnt mit der nach SYNC folgenden steigenden Flanke von $\overline{R \times C}$. Zwischen 2 aktiven SYNC-Signalen müssen mindestens 3 Takte liegen. Im Asynchronmodus sind diese Anschlüsse direkte Eingänge für das HUNT SYNC-Bit im Statusregister RR0.

2.4.3. Aufbau eines Kanals des *U 880* — SIO

Bild 2.9 zeigt den Aufbau eines Kanals. Zu einem Kanal gehören folgende Register:
Verschieberegister für den Empfang,
Verschieberegister für das Senden,

* Aus Gründen begrenzter Pin-Anzahl stehen für $\overline{T \times CB}$; $\overline{R \times CB}$ und $\overline{DTRB}$ nur 2 Pins zur Verfügung.
Standardversion: $\overline{T \times CB}$ und $\overline{R \times CB}$ gleich 1 Pin $\overline{R \times T \times CB}$.
Sonderversion: $\overline{T \times CB}$, $\overline{R \times CB}$ 2 Pins, DTRB fehlt.

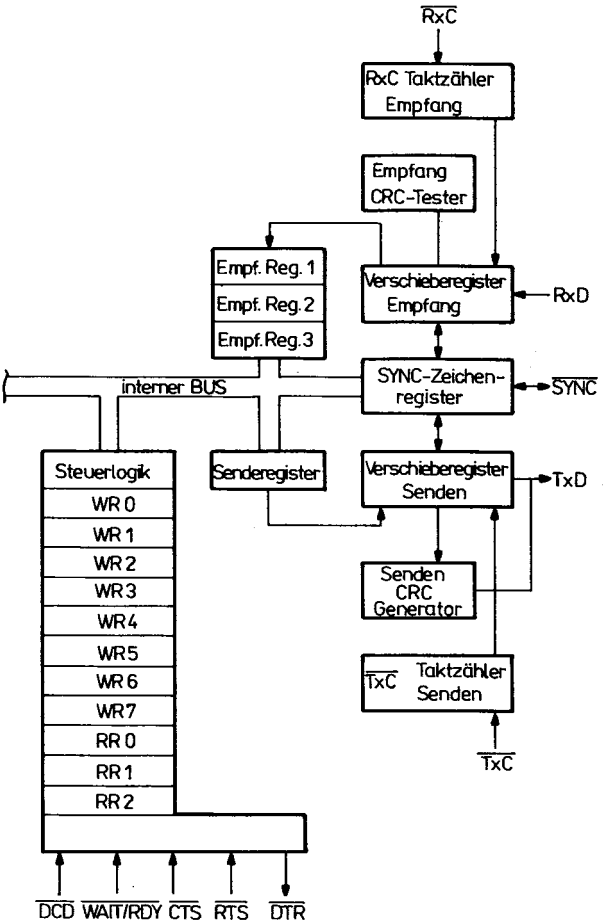


Bild 2.9 Logische Struktur eines Kanals des Bausteins *U 880* — SIO

drei Empfangspufferregister,
Senderpufferregister,
Register für die Speicherung des SYNC-Zeichens,
acht Register für Steuerinformationen (WR_0 bis WR_7),
drei Register für Statusinformationen (RR_0 bis RR_2).

Senden von seriellen Daten:

Die zu sendende Information gelangt zuerst in den Sendepuffer, anschließend kommt sie in das Verschieberegister und wird von da aus seriell über den Ausgang T × D gesendet. Die Sende-Bit erscheinen mit der fallenden Flanke von $\overline{T \times C}$, wobei durch den „Taktzähler Senden" eine Untersetzung programmierbar ist.

Empfang von seriellen Daten:

Die Daten kommen über den Eingang R × D ins Verschieberegister und von da aus in die 3 Empfängerpuffer. Die Eingangsdaten werden mit der steigenden Flanke von $\overline{R \times C}$ abgetastet. Durch den „Taktzähler Empfang" läßt sich eine Untersetzung des Empfangstaktes programmieren.

Programmierung des SIO:

Die Initialisierung des SIO geschieht mit Hilfe von Steuerbytes ($C/\overline{D}$-Pin = „High"). Das 1. Steuerbyte nach RESET geht in das Register WR_0. Das 2. Steuerbyte wird in ein WR-Register gespeichert, dessen Nummer in den Bit-Stellen D_2, D_1, D_0 von WR_0 steht. Das nächste Steuerbyte geht wieder nach WR_0, dann wieder in das durch die Bit-Stellen D_2, D_1, D_0 angezeigte Register und so fort. Beim Lesen der RR-Register muß nach RESET zunächst ein Steuerbyte in das Register WR_0 gebracht werden, in dem die Nummer des zu lesenden RR-Registers angegeben ist.

2.4.4. Arbeitsweise des SIO-Bausteins

Asynchronmodus

Sendeformat

Nullinie								Nullinie
BREAK	Start	D_0	D_1	D_2	...	D_n	Parity	STOP
Programmierbar ja/nein		5, 6, 7, 8 Bit/Zeichen					Programmierbar ja/nein gerade/ ungerade	1, 1 1/2 oder 2 Bit

Senden
- Senden beginnt mit TRANSMIT ENABLE
- bei AUTOENABLE muß $\overline{\text{CTS}}$ „Low" sein,
- bei n < 8 Bit müssen vordere Bit 0 sein.

Empfang
- Empfang beginnt mit RECEIVER ENABLE,
- bei AUTOENABLE muß $\overline{\text{DCD}}$ „Low" sein.

Die empfangenen Bit werden in der Mitte einer Bit-Zeit abgetastet. Ist 1/2 Bit-Zeit nach der fallenden Flanke des Eingangssignals noch Tiefpegel, so wird das als Start-Bit erkannt, und die folgenden Bit werden empfangen.
- BREAK kann durch Setzen eines Bit in einem WR-Register gesendet werden. Es entsteht „Low"-Pegel, solange das Bit gesetzt ist.

Synchronmodus

Im Synchronmodus beginnt der Empfang erst nach Erkennen der Synchronisation. Vom Sender kommt entweder ein SYNC-Signal (externe Synchronisation) oder ein Synchronisationszeichen im Text (interne Synchronisation). Im Synchronmodus existiert nach RESET und RECEIVER ENABLE der HUNTMODE. In diesem Zustand werden keine Daten empfangen. Der HUNTMODE wird beendet, wenn Synchronisation eingetreten ist. Danach beginnt der Datentransfer. Ist die Synchronisation verlorengegangen, so kann der HUNTMODE gesetzt werden.

Monosynchronisation

Der Empfang beginnt, wenn das SYNC-Zeichen (8 Bit) erkannt ist. Das Vergleichszeichen steht in WR_7. Bei Erkennen des SYNC-Zeichens wird der $\overline{\text{SYNC}}$-PIN aktiv (bis zum Ende des Taktzyklus, in dem das SYNC-Zeichen erkannt wurde).

Bisynchronisation

Der Empfang beginnt, wenn ein 16-Bit-SYNC-Wort (2 aufeinanderfolgende Zeichen) erkannt ist. Die Vergleichszeichen stehen in WR_6 und WR_7. Bei Erkennen der SYNC-Zeichen wird der $\overline{\text{SYNC}}$-PIN aktiv (vom Erkennen bis zum Ende des Taktzyklus, in dem es erkannt wurde).

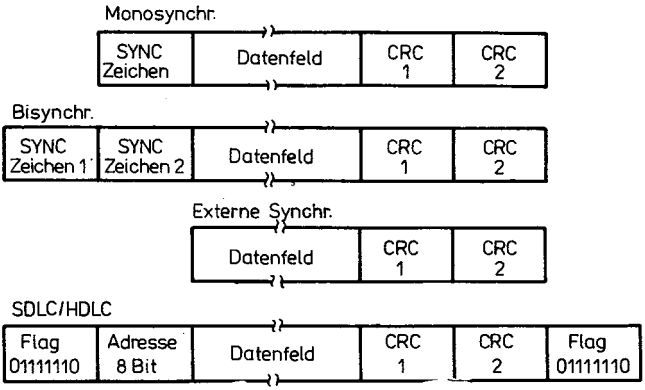


Bild 2.10 Datenformate im Synchronmodus des Bausteins *U 880* — SIO

Externe Synchronisation

Der Empfang beginnt mit der 1. steigenden Flanke von $\overline{R \times C}$, nachdem $\overline{SYNC}$ ($\overline{SYNC}$-PIN) aktiv geworden ist.
Bild 2.10 zeigt die Datenformate für die Betriebsarten des Synchronmodus.

Synchronmodus (Zusammenfassung)

Sender

- Mit der fallenden Flanke von $\overline{T \times C}$ wird ein Bit gesendet.
- Grundzustand (nach RESET) ist Markierungslinie.
- Nach Modusauswahl und TRANSMIT ENABLE ist Grundzustand „Senden von Synchronisationszeichen".
- BREAK ergibt Grundpegel („Low-Pegel")

Empfänger

- Mit der steigenden Flanke von $\overline{R \times C}$ wird die Information abgetastet.
- Der Empfänger ist nach RESET im HUNTMODE. Er beginnt mit RECEIVER ENABLE.
- Nach Synchronisation beginnt die Datenübernahme.
- Synchronisation liegt vor, wenn
  - bei Monosynchronisation das das Zeichen WR_7 erkannt ist;
  - bei Bisynchronisation

das 16-Bit-Wort in WR_6 und WR_7 erkannt ist;
- bei externer Synchronisation der $\overline{\text{SYNC}}$-PIN von außen „Low" gesetzt ist.

– Abbruch des Empfangs erfolgt, wenn:
- RESET erscheint;
- Bit RECEIVER ENABLE rückgesetzt wird;
- im AUTOENABLE-Zustand $\overline{\text{DCD}}$ high wird;
- der HUNTMODE gesetzt wird.

CRC-Bildung

– CRC-Berechnung beginnt 8 Bit-Zeiten, nachdem das Wort im Empfängerpuffer ist.
– CRC Codes: $x^{16} + x^{15} + x^2 + 1$ bzw.
$x^{16} + x^{12} + x^5 + 1$ bei SDLC.
– CRC-Berechnung läßt sich sperren und freigeben durch das Bit TRANSMIT CRC ENABLE.
– Während CRC-Sendung ist das CRC/SYNC-Bit gesetzt. Das TRANSMIT-BUFFER-EMPTY-Bit ist gesetzt.
– SYNC-Zeichen gehen nicht in CRC ein.

2.4.5. Interruptbildung und Statusbildung

Es gibt 4 Hauptinterruptursachen:

1. Es kommt zum Interrupt, wenn der Sendepuffer keine Daten mehr zum Senden hat. Dieser Interrupt kann durch Nullsetzen des Bit TRANS INTERRUPT ENABLE im Steuerrgeister WR_1 gesperrt werden.
2. Externer Interrupt bzw. Interrupt tritt auf bei Statuswechsel. Hierzu gehören 5 Ursachen, die im Statusregister RR_0 durch die entsprechenden Bit angezeigt werden:
   – Zustandswechsel („Low"-„High" oder umgekehrt) am CTS-Eingang.
   – Zustandswechsel am DCD-Eingang.
   – Zustandswechsel am SYNC-Eingang (Asynchronmodus und

Synchronmodus mit externer Synchronisation) bzw. am SYNC-Ausgang (Monosyncmodus und Bisyncmodus).
- BREAK oder eine Abbruch-Bit-Folge ist erkannt worden.
- Beginn des Sendens des CRC-Wortes.

Dieser Interrupt kann durch Nullsetzen des Bit EXT INTERRUPT ENABLE im Steuerregister WR_1 gesperrt werden.

3. Je nach eingestelltem Interruptmodus tritt ein Interrupt dann auf, wenn entweder das 1. Zeichen nach Einstellung des Interruptmodus in den Empfängerpuffer gelangt ist oder wenn mindestens 1 Zeichen im Empfängerpuffer zur Verfügung steht. Durch Nullsetzen der Bit D_3 und D_4 im Steuerregister WR_1 kann dieser Interrupt gesperrt werden.
4. Zum Interrupt kommt es bei einer besonderen Empfangsbedingung. Hierzu gehören 4 Ursachen, die im Statusregister RR_1 durch die entsprechenden Bit angezeigt werden:
   - Paritätsfehler,
   - Empfängerpufferüberlauf,
   - Formatfehler,
   - Blockende (nur SDLC).

Wann diese Interrupt zugelassen oder gesperrt sind, hängt vom eingestellten Interruptmodus ab.
Nach jedem Interrupt wird die Hauptinterruptursache in den Interruptvektor (Steuerregister WR_2) Bit 1 und 2 eingetragen, sofern das Bit STATUS AFFECTS VECTOR im Steuerregister WR_1 gesetzt ist.

Interruptbildung Senden
- TRANSMIT BUFFER EMPTY (bei ASYNC):
  Interrupt tritt auf, wenn der Sendepuffer gerade leer geworden ist, d. h., wenn keine zu sendenden Daten mehr im Sendepuffer sind.
- CTS (bei ASYNC und SYNC):
  Es kommt zum externen Interrupt, wenn ein Zustandswechsel („Low"-„High" oder umgekehrt) am CTS-Eingang aufgetreten ist.
- CRC/SYNC (bei SYNC):
  Während des Sendens des CRC-Wortes ist das Bit TRANSMIT BUFFER EMPTY noch nicht gesetzt. Nachdem das CRC-Wort gesendet ist, wird das Bit TRANSMIT BUFFER EMPTY gesetzt, das wiederum einen Interrupt erzeugt. Wenn das CRC-

Senden beginnt, wird das Bit SENDING CRC/SYNC gesetzt und ein Statusinterrupt erzeugt.

- TRANSMIT BUFFER EMPTY (bei SYNC):
  Wenn der CRC-Generator nicht arbeitet, kommt es zum Interrupt, wenn der Sendepuffer leer geworden und das 1. SYNC-Zeichen in den Sendepuffer geladen ist. Es beginnt das automatische Senden von SYNC-Zeichen (Monosyncronmodus, Bisyncronmodus).
  Interrupt tritt auf, wenn der CRC-Generator arbeitet und wenn der Sendepuffer leer geworden sowie das CRC-Wort gesendet worden ist. Dann beginnt das automatische Senden von SYNC-Zeichen.

Interruptbildung Empfang

Interruptmodus 1

- RECEIVE CHARACTER AVAILABLE (bei ASYNC und SYNC):
  Interrupt tritt auf, wenn das 1. Zeichen eines zu übertragenden Blockes, d. h. das 1. Zeichen nach Einstellen des Interruptmodus, im Empfängerpuffer angekommen ist.
- SYNC/HUNT (bei SYNC):
  Es kommt zum externen Interrupt bei einem Zustandswechsel am SYNC-Ausgang vom Zustand „Zeichensynchronisation erreicht" in den Zustand „Herstellen Zeichensynchronisation" und umgekehrt.
- SYNC/HUNT (bei ASYNC):
  Es kommt zum Statusinterrupt, wenn ein Zustandswechsel am SYNC-Eingang aufgetreten ist.
- BREAK/ABORT (bei ASYNC):
  Ein Statusinterrupt tritt auf, wenn ein Break erkannt worden ist.
- DCD (bei ASYNC und SYNC):
  Es kommt zum externen Interrupt, wenn ein Zustandswechsel am DCD-Eingang aufgetreten ist.
- Besondere Empfangsbedingung:
  Interrupt tritt auf bei

- Empfängerpufferüberlauf (bei ASYNC und SYNC),
- Formatfehler (bei ASYNC).

  Paritätsfehler führen nicht zum Interrupt.

Interruptmodus 2

- RECEIVE CHARACTER AVAILABLE (bei ASYNC und SYNC):
  Interrupt tritt auf, wenn mindestens 1 Zeichen im Empfängerpuffer verfügbar ist. Dieser Interrupt kommt bei jedem Zeichen.
- DCD:
  wie Interruptmodus 1
- SYNC/HUNT:
  wie Interruptmodus 1
- Besondere Empfangsbedingung:
  Interrupt bei Paritätsfehler (bei ASYNC und SYNC).
  Der Paritätsfehlerinterrupt ist mit einer Eintragung im Interruptvektor verbunden.

Interruptmodus 3

Wie Interruptmodus 2, jedoch verursacht der Paritätsfehlerinterrupt keine Eintragung im Interruptvektor.

Statusbildung Senden

- TRANSMIT BUFFER EMPTY (bei ASYNC):
  Das Bit wird gesetzt, wenn der Sendepuffer leer geworden ist. Es bleibt gesetzt, solange der Sendepuffer leer ist.
- TRANSMIT BUFFER EMPTY (bei SYNC):
  Bei nicht arbeitendem CRC-Generator wird das Bit gesetzt, wenn der Sendepuffer leer geworden ist, d. h., wenn keine zu sendenden Zeichen mehr im Sendepuffer sind. Es wird mit dem automatischen Senden von SYNC-Zeichen begonnen (Monosyncmodus, Bisyncmodus). Das Bit ist gesetzt, solange der Sendepuffer leer ist, d. h., solange er keine Daten zum Senden hat.
  Bei arbeitendem CRC-Generator wird das Bit gesetzt, wenn der Sendepuffer leer geworden und das CRC-Wort gesendet worden ist. Es beginnt das automatische Senden von SYNC-Zeichen
- (Monosyncmodus, Bisyncmodus). Wenn man den CRC-Generator nicht abschaltet, so wird über die SYNC-Zeichen ein neues Wort berechnet. Das Bit bleibt so lange gesetzt, bis der Sendepuffer leer ist, d. h., solange er keine Daten zum Senden hat.
- CTS (bei ASYNC und SYNC):
  Das Bit wird gesetzt entsprechend dem Zustand des CTS-Ein-

gangs zum Zeitpunkt des letzten Wechsels eines der 5 Status DCD, CTS, SYNC/HUNT, BREAK/ABORT oder SENDING CRC/SYNC.

- ALL SENT (bei ASYNC):
  Das Bit wird gesetzt, wenn alle Bit des Schieberegisters den Sender verlassen haben. Es verursacht keinen Interrupt.
- CRC/SYNC (bei SYNC):
  Das Bit wird gesetzt, wenn, nachdem der Sendepuffer leer geworden ist, mit dem Senden des CRC-Wortes begonnen wird. Nachdem Senden des CRC-Wortes wird das Bit TRANSMIT BUFFER EMPTY gesetzt und mit dem Senden von SYNC-Zeichen begonnen (Monosyncmodus, Bisyncmodus). Das Bit bleibt so lange gesetzt, bis es durch das Kommando „RESET CRC/SYNC, RESET EXTERNAL/STATUS INTERRUPT oder CHANNEL RESET zurückgesetzt wird. Voraussetzung für diese Statusanzeige ist, daß der CRC-Generator arbeitet.

Bit CRC/SYNC gesetzt und Bit TRANSMIT BUFFER EMPTY nicht gesetzt	CRC-Wort wird gesendet
Bit CRC/SYNC gesetzt und Bit TRANSMIT BUFFER EMPTY gesetzt	SYNC-Zeichen werden gesendet

Statusbildung Empfang

- RECEIVER CHARACTER AVAILABLE (ASYNC und SYNC):
  Das Bit ist so lange gesetzt, wie mindestens 1 Zeichen im Empfängerpuffer verfügbar ist.
- DCD (ASYNC und SYNC):
  Das Bit wird entsprechend dem Zustand des DCD-Signals zum Zeitpunkt des letzten Wechsels eines der 5 Status DCD, CTS, SYNC/HUNT, BREAK/ABORT oder CSC/SYNC gesetzt.
- SYNC/HUNT (ASYNC):
  Das Bit wird entsprechend dem Zustand am SYNC-Eingang zum Zeitpunkt des letzten Wechsels eines der 5 Status DCD, CTS, SYNC/HUNT, BREAK/ABORT oder CRC/SYNC gesetzt.
- SYNC/HUNT (SYNC):
  Das Bit wird gesetzt, wenn durch Setzen des Bit ENTER HUNT-MODE der Zustand „Herstellen Zeichensynchronisation" ein-

gestellt wird. Es wird rückgesetzt, wenn die Zeichensynchronisation erreicht ist, und bleibt rückgesetzt, auch wenn die Zeichensynchronisation abbricht, bis zum erneuten Setzen des Bit ENTER HUNTMODE. Bei abgeschaltetem Empfänger oder nach einem CHANNEL-RESET-Kommando ist dieses Bit gesetzt.

INTERRUPT PENDING (ASYNC und SYNC):
Dieses Bit wird gesetzt, wenn irgendeine Interruptbedingung im gesamten SIO vorhanden ist. Es existiert nur im Kanal A.

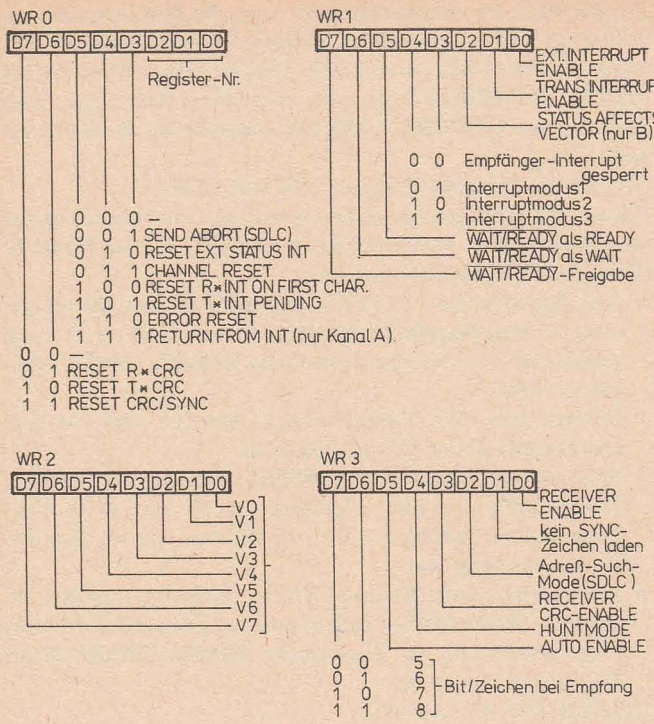
- Besondere Empfangsbedingungen:
  - Empfängerüberlauf (ASYNC und SYNC): Das Bit ist gesetzt bei Empfängerpufferüberlauf. Danach bleibt es so lange gesetzt, bis es durch das Kommando ERROR RESET rückgesetzt wird.
  - Format- oder CRC-Fehler (SYNC): Das Bit zeigt das Vergleichsergebnis der CRC-Prüfung an.
  - Format- oder CRC-Fehler (ASYNC): Das Bit wird gesetzt, wenn ein Formatfehler aufgetreten ist. Es wird nicht blokkiert, sondern bleibt nur für das Zeichen gesetzt, das den Fehler ausgelöst hat.
  - Paritätsfehler (ASYNC und SYNC): Dieses Bit wird gesetzt, wenn ein Paritätsfehler aufgetreten ist. Danach bleibt es so lange gesetzt, bis das Kommando ERROR RESET dieses Bit wieder rücksetzt.
- BREAK/ABORT (ASYNC):
  Das Bit wird gesetzt, wenn ein Break erkannt worden ist. Erst wenn dieser Break-Zustand nicht mehr besteht, kann dieses Bit durch das Kommando RESET EXTERNAL/STATUS INTERRUPT rückgesetzt werden.

2.4.6. Beschreibung der Bit-Stellen der Steuerregister WR_0 bis WR_7 und der Statusregister RR_0 bis RR_2 (s. Bild 2.11)

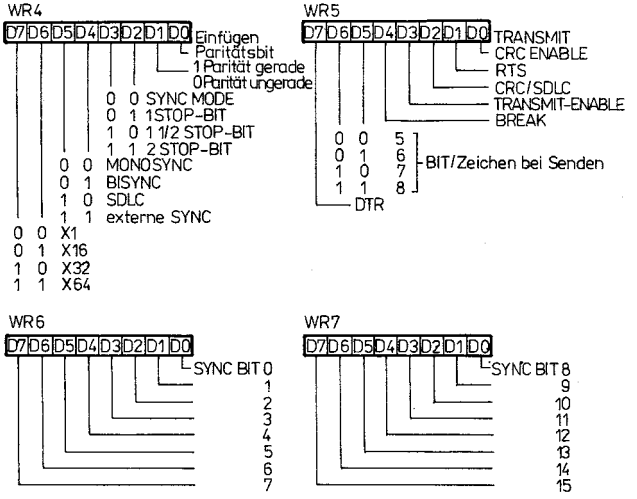
WR_0:

$D_0 \ldots D_2$ Nr. des nächsten zu ladenden oder zu lesenden Registers.

$D_3 \ldots D_5$ RESET-Kommandos
SEND ABORT (nur SDLC). Es wird eine Folge von 8 bis 13 „1" gesendet.



RESET EXT STATUS INT löscht die im RR_0 bei Statusinterrupt eingetragenen Bit.
CHANNEL RESET führt zum Rücksetzen eines Kanals. Alle Steuer- und Statusregister sind gelöscht. RESET R × INT ON FIRST CHAR. Nach Erscheinen eines Interrupt im Interruptmodus 1 muß durch RESET R × INT ON FIRST CHAR der Kanal für weitere Interrupt freigegeben werden. RESET T × INT PENDING. Im Interruptmodus 2 erscheint bei leerem Sendepuffer ein Interrupt. Dieses Kommando verhindert, daß dieser Interrupt mehrmals kommt. Nach dem Kommando tritt so



lange kein Interrupt auf, bis wieder ein Zeichen in den Sendepuffer gekommen ist. ERROR RESET veranlaßt das Rücksetzen der Fehler-Bit (Parität und Empfängerüberlauf) in RR_1.
RETURN FROM INT erzeugt einen RETI-Befehl auf dem Datenbus.

$D_6 \ldots D_7$ setzt die CRC-Logik zurück.

W R_1:

D_0 EXT INTERRUPT ENABLE. Gestattet Interrupt bei Wechsel von $\overline{DCD}$, $\overline{CTS}$, $\overline{SYNC}$, bei BREAK, bei Sendebeginn von CRC oder SYNC-Zeichen.

D_1 TRANS INTERRUPT ENABLE. Erlaubt INT, wenn der Sendepuffer leer ist.

D_2 STATUS AFFECTS VECTOR (nur Kanal B). Ist dieser Modus ausgewählt, so wird der Interruptvektor in WR_2 wie folgt modifiziert:

V_3	V_2	V_1		
0	0	0	Sendepuffer leer	Kanal B
0	0	1	externer Statuswechsel	
0	1	0	empfangenes Zeichen verfügbar	
0	1	1	Kein INTERRUPT	

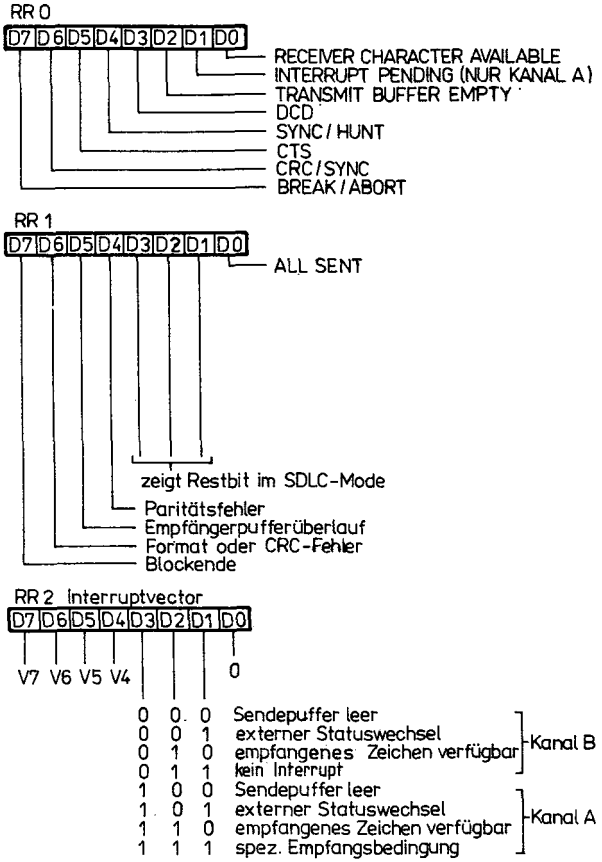


Bild 2.11 Bedeutung der Bitstellen der Steuerregister WR_0 bis WR_7 und der Statusregister RR_0 bis RR_2 im Baustein *U 880* — SIO

1	0	0	Sendepuffer leer	Kanal A
1	0	1	externer Statuswechsel	
1	1	0	empfangenes Zeichen verfügbar	
1	1	1	spezielle Empfangsbedingung*)	

*) spezielle Empfangsbedingung: Paritätsfehler,

Empfängerüberlauf,
CRC-/Format-Fehler,
Blockende
(SDLC).

$D_3 \ldots D_4$ Durch D_3 und D_4 werden spezielle Interruptbildungsmöglichkeiten für Empfänger eingestellt.

D_5 $D_5 = 1$ Der $\overline{\text{WAIT-/READY}}$-Ausgang wird aktiv, wenn der Empfänger leer ist.

$D_5 = 0$ Der $\overline{\text{WAIT-/READY}}$-Ausgang wird aktiv, wenn der Sendepuffer voll ist.

D_6 $D_6 = 0$ $\overline{\text{WAIT-/READY}}$-Ausgang arbeitet als WAIT (Ausgang).

$D_6 = 1$ $\overline{\text{WAIT-/READY}}$-Ausgang arbeitet als READY (Ausgang).

WAIT-Funktion: Das SIO kann keine Daten aufnehmen oder abgeben.

READY-Funktion: Das SIO ist zur Arbeit bereit.

D_7 Freigabe des WAIT-/READY-Ausgangs.

WR_2: Dieses Register enthält den Interruptvektor. Er existiert nur im Kanal B. Bei STATUS AFFECTS VECTOR werden die Bit-Stellen V_1 bis V_3 entsprechend der Interrupterzeugung eingetragen.

WR_3:

D_0 Erlaubt Empfangsfunktion.

D_1 Bei $D_1 = 1$ werden die SYNC-Zeichen nicht in die Empfangspufferregister geladen. CRC erfolgt normal.

D_2 $D_2 = 1$ verhindert im SDLC-Modus Interrupt, wenn keine Übereinstimmung zwischen empfangener und programmierter Adresse oder wenn keine Globaladresse (11111111) vorliegt.

D_3 Mit $D_3 = 1$ beginnt die CRC-Kontrolle beim Start des letzten Zeichens vom Empfangsregister zum Puffer.

D_4 HUNTMODE. Die Übertragung wird abgebrochen, bis ein neues SYNC-Zeichen kommt.

D_5 AUTOENABLE. Bei $D_5 = 1$ arbeiten die DCD- und CTS-Eingänge als Empfangs- und Sendesteuerung.

Bei $D_5 = 0$ wirken DCD und CTS nur auf ihre Bit im Statusregister.

D_6/D_7 Bestimmt empfangene Bit/Zeichen.

WR_4:

D_0 $D_0 = 1$, SIO arbeitet mit Paritäts-Bit,

D_1 $D_1 = 0$, Parität gerade,
$D_1 = 1$, Parität ungerade.

D_2/D_3 Bestimmt die Anzahl der Stop-Bit bei Senden.

D_4/D_5 Bestimmt die Art des Synchronmodus.

D_6/D_7 Bestimmung des Multiplikationsfaktors zwischen Takt und Übertragungsrate.

WR_5:

D_0 TRANSMIT CRC-ENABLE. CRC-Freigabe bei Senden

D_1 $D_1 = 1$, $\overline{\text{RTS}}$ geht auf 0.
$D_1 = 0$, $\overline{\text{RTS}}$ geht erst dann auf 1, wenn der Sender leer ist.

D_2 CRC/SDLC
$D_2 = 0$, CRC-Kontrolle mit SDLC-Polynom
$x^{16} + x^{12} + x^5 + 1$.
$D_2 = 1$, CRC-Kontrolle mit Polynom
$x^{16} + x^{15} + x^2 + 1$.

D_3 TRANSMIT ENABLE.
Sendeerlaubnis. Bei Rücksetzen wird das letzte Zeichen noch voll ausgegeben.

D_4 BREAK. Bewirkt Aussenden von „0" auf T × D.

D_5/D_6 Bestimmt auszugebende Bit/Zeichen.
Bei weniger als 5 Bit/Zeichen gilt:

1	1	1	1	0	0	0	D	sendet 1 Bit
1	1	1	0	0	0	D	D	sendet 2 Bit
1	1	0	0	0	D	D	D	sendet 3 Bit
1	0	0	0	D	D	D	D	sendet 4 Bit
0	0	0	D	D	D	D	D	sendet 5 Bit

D_7 DTR
$D_7 = 1$ bewirkt $\overline{\text{DTR}} = 0$ (aktiv).

WR_6: SYNC-Zeichen bei Monosynchronisation, die ersten 8 Bit bei Bisynchronisation.

WR_7: 2. Zeichen bei Bisynchronisation oder das Synchron-

zeichen bei Monosynchronisation. Bei SDLC muß $WR_7 = 01111110$ sein.

RR_0:

D_0 RECEIVE CHARACTER AVAILABLE; gesetzt, wenn mindestens 1 Zeichen im Puffer.

D_1 INTERRUPT PENDING; wird bei jeder INT-Anmeldung gesetzt (nur Kanal A).

D_2 TRANSMIT BUFFER EMPTY; gesetzt ,wenn der Sendepuffer leer ist (außer bei CRC).

D_3 DCD; zeigt Zustand $\overline{DCD}$.

D_4 SYNC/HUNT
$D_4 = 0$, Zeichensynchronisation erreicht,
$D_4 = 1$, keine Synchronisation.

D_5 CTS; zeigt Zustand $\overline{CTS}$.

D_6 BREAK/ABORT; wird gesetzt, wenn mit dem Senden von CRC-Worten begonnen wird, nachdem der Sendepuffer leer geworden ist.

D_7 CRC/SYNC
$D_7 = 1$ bei Erkennen von BREAK (bei ASYNC). Im SDLC-Format bei Erkennen einer Abbruchfolge (7 oder mehr Einsen).

RR_1:

D_0 ALL SENT wird gesetzt, wenn alle Zeichen im Asynchronmode gesendet sind.

$D_1/D_2/D_3$ Falls bei Empfang im SDLC-Modus die Anzahl der Bit/Zeichen nicht aufgehen, zeigt $D_1/D_2/D_3$ die restlichen Bit an.

D_4 Paritätsfehler,

D_5 Empfängerüberlauf,

D_6 Formatfehler im Asynchronmode,
CRC-Fehler im Synchronmode.

D_7 Blockende bei SDLC mit gültigem CRC erkannt.

RR_2: Enthält Interruptvektor. Die Bit sind entsprechend STATUS AFFECTS VECTOR verändert.

2.5. DMA-Steuerbaustein *U 880* — DMA

2.5.1. Funktion und Struktur des DMA-Bausteins

Der DMA-Baustein dient zur Steuerung der Übertragung von

Daten von einer Funktionseinheit des Mikrorechners zu einer anderen Funktionseinheit. Als Funktionseinheiten können dabei periphere Einheiten oder der Arbeitsspeicher auftreten. Damit ist es möglich, mit Hilfe des DMA folgende Datenübertragungen durchzuführen:

- von einer peripheren Einheit zu einer anderen;
- von einem Bereich im Arbeitsspeicher zu einem anderen,
- von einer peripheren Einheit zum Arbeitsspeicher,
- vom Arbeitsspeicher zu einer peripheren Einheit.

Die Übertragung geschieht blockweise. Die Übertragungsgeschwindigkeit beträgt bis zu 1,25 Megabyte/s.

Bild 2.12 zeigt die Grundstruktur des DMA-Bausteins. Der Baustein hat folgende Funktionseinheiten, die über einen internen Bus gekoppelt sind:

- 1 Interface mit Steuerung zum Systembus des Mikrorechners,
- 2 Tore; sie werden mit A und B gekennzeichnet.
  Jedes dieser Tore kann die Steuerung einer Funktionseinheit (periphere Einheit oder Arbeitsspeicher) übernehmen. Es kann zum Lesen und zum Einschreiben benutzt werden.
- 1 Steuerlogik mit einem Registersatz,
- 1 Interrupt- und Bussteuerung.

Der DMA-Baustein hat folgende Register:

Steuerregister	Dieses Register enthält die notwendigen Steuerinformationen.
Zeitablaufregister	In diesem Register ist die Steuerinformation für die Art und Weise der Zyklensteuerung enthalten.
Interruptvektorregister	Dieses Register enthält den Interruptvektor.
Blocklängenregister	Dieses Register enthält die gesamte zu suchende oder zu übertragende Blocklänge.
Bytezähler	Die gelesenen Bytes werden im Bytezähler gezählt. Bei Übereinstimmung mit dem Blocklängenregister wird ein Bit im Statusregister gesetzt.
Vergleichsregister	Es enthält ein Byte, nach dem beim Suchvorgang gesucht wird.
Maskenregister	Dieses Register enthält eine Maske. Ein Vergleich eines Bit zwischen gelesenem Byte und Vergleichsbyte findet statt, wenn

	im Maskenregister an der entsprechenden Stelle eine „0" steht.
Startadreßregister für Tor A und B	Sie enthalten die Startadressen (16 Bit) für die beiden Tore. Bei E/A-Operationen sind es nur 8 Bit.
Adreßzähler für Tor A und B	Diese Zähler enthalten die laufenden Adressen der beiden Tore (16 Bit).
Pulssteuerregister	Sobald die im Pulssteuerregister angegebene Anzahl Byte abgearbeitet ist, wird am Ausgang $\overline{INT}$ ein Impuls ausgegeben. Die CPU reagiert nicht, da durch $\overline{BUSRQ}$ der DMA aktiviert ist.
Statusregister	Dieses Register enthält die gesammelte Statusinformation.

2.5.2. Beschreibung der Anschlüsse des DMA-Bausteins

$A_0 \ldots A_{15}$	Adreßbus (Tri-state-Ausgang, aktiv, „High");	
$D_0 \ldots D_7$	Datenbus (Tri-state, bidirectional, aktiv „High");	
$\overline{M_1}$	Signal für M_1-Zyklus (Eingabesignal, aktiv „Low");	
$\overline{IORQ}$	Eingabe-/Ausgabe-Anforderung (Ein- bzw. Ausgabesignal, aktiv „Low");	
$\overline{MREQ}$	Speicheranforderung	(Ein- bzw. Ausgang, aktiv „Low");
$\overline{RD}$	Lesen	(Ein- bzw. Ausgang, aktiv „Low");
$\overline{WR}$	Schreiben	(Ein- bzw. Ausgang, aktiv „Low");
$\overline{CE/WAIT}$		(Eingang, aktiv „Low"); dient als Bausteinfreigabesignal. Es kann, wenn $\overline{BAI}$ = „Low" ist, auch als „WAIT" verwendet werden. Die Auswahl erfolgt durch Programm.
$\overline{BUSRQ}$	Busanforderung	(Ein-/Ausgang, aktiv „Low"); Signal zur Anforderung einer DMA-Funktion.

$\overline{\text{BAI}}$	Busanforderungsbestätigung	(Eingang, aktiv „Low"); zeigt an, daß die Buskontrolle an den DMA-Baustein übergeben wurde.
$\overline{\text{BAO}}$	Weitergabe $\overline{\text{BAI}}$	(Ausgang, aktiv „Low"); gibt $\overline{\text{BAI}}$-Signal an den nächsten Baustein weiter.
$\overline{\text{INT}}$	Interruptanforderung	(Ausgang, aktiv „Low");
IEI	Interruptfreigabe	(Eingang, aktiv „High");
IEO	Weitergabe Interruptfreigabe	(Ausgang, aktiv „High").
RDY	Quittung	(Eingang aktiv „High" oder „Low", je nach Programmierung). RDY teilt dem DMA mit, ob die periphere Einheit zum Lesen oder Schreiben bereit ist.

2.5.3. Arbeitsweise des DMA-Bausteins

Busanforderung

Wenn RDY aktiv ist, erzeugt der DMA das Signal $\overline{\text{BUSRQ}}$. Die CPU aktiviert danach BUSAK. Mit BUSAK muß BAI aktiv werden. Nach BAI beginnt mit der nächsten Taktflanke von Φ ein DMA-Zyklus. Nach der Busanforderung läuft die eingestellte Betriebsart ab.

Betriebsarten

Alle Betriebsarten können in den Funktionsarten Übertragung, Suchen, Übertragung und Suchen laufen. Es gibt folgende Betriebsarten:

Blockweise	Die Übertragung ist abgeschlossen, wenn der Block übertragen oder das gesuchte Byte gefunden ist (unabhängig von RDY).
andauernd (burst)	Die Blockübertragung arbeitet nur, wenn RDY aktiv ist.
transparent	Die DMA-Operation wird während der Refresh-Zyklen ausgeführt.
Byteweise	Die CPU übernimmt die Buskontrolle nach jeder Einzelbyteoperation.

Blockübertragung

Bei Speicherzugriff steuert der DMA einen Speicherlesezyklus und speichert das Byte im Lesetor. Danach erfolgen bei „Übertragung" die Übergabe zum Schreibtor und ein Speicherschreibzyklus. Bei „Suchen" kommt es zum Vergleich mit dem Byte im Vergleicherregister, und bei Gleichheit wird ein Status-Bit gesetzt. Adreß- und Bytezähler werden dabei erhöht. Bei Ein-/Ausgabe laufen Ein- bzw. Ausgabezyklen ab. Die Zyklenlänge läßt sich programmieren, jedoch muß die Mindestlänge analog den CPU-Zyklen eingehalten werden. Durch die Verlängerung spart man eine zusätzliche WAIT-Logik. Nach RESET arbeiten die Zyklen analog den CPU-Zyklen.

Die Busfreigabe geschieht bei „Übertragung" nur am Blockende, bei „Suchen" bei Gleichheit oder am Blockende. Bei Gleichheit wird vor der Busfreigabe noch eine Byteoperation ausgeführt.

Andauernde Übertragung

Der Ablauf in dieser Betriebsart unterscheidet sich von der „Blockübertragung" dadurch, daß es während RDY = „0" zu einer Busfreigabe kommt.

Byteweise Übertragung

Die Busfreigabe ist unabhängig von RDY nach jedem Lesezyklus bei „Suchen" und nach jedem Schreibzyklus bei „Übertragen". Nachdem $\overline{\text{BUSRQ}}$ inaktiv ist, kann ein erneutes $\overline{\text{BUSRQ}}$ erst kommen, wenn vorher $\overline{\text{BAI}}$ inaktiv geworden ist.

Programmierung des DMA (Initialisierung)

Das Einstellen der Funktion des DMA geschieht durch eine Reihe Steuerbytes, mit deren Hilfe die einzelnen Register gefüllt werden. Die unterschiedlichen Steuerbytes sind durch die Bit-Stellen D_0, D_1 und D_7 gekennzeichnet. Die Steuerbytes werden in Steuerregister gespeichert. Sie werden 1A, 1B, 2A, 2B, 2C, 2D genannt.

Kennzeichnung der Steuerbytes:

	D_7	D_1	D_0
1A	0	± 0	
1B	0	0	0
2A	1	0	0
2B	1	0	1
2C	1	1	0
2D	1	1	1

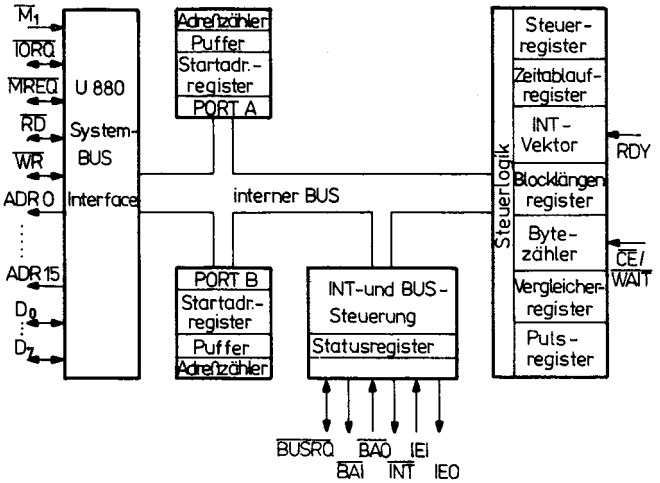


Bild 2.12 Grundstruktur des Bausteins *U 880* — DMA

Übersicht über die Steuerbytes

Steuerbyte 1A

D_7	0
D_6	Blocklänge, „High"-Byte folgt
D_5	Blocklänge, „Low-"Byte folgt
D_4	Adresse A, „High"-Byte folgt
D_3	Adresse A, „Low"-Byte folgt
D_2	0 B → A; 1 A → B
D_1	0 — ; 0 1 Transfer; 1 0 Suche; 1 1 Suche + Transfer
D_0	

Steuerbyte 1B

D_7	0
D_6	Zeitablaufbyte folgt
D_5	Toradresse ist fest
D_4	0 Toradresse wird decrementiert 1 Toradresse wird incrementiert
D_3	0 0 B = Speicher; 0 1 A = Speicher; 1 0 B = E/A;
D_2	1 1 A = E/A

D_1 0
D_0 0

Zeitablaufbyte

D_7 0 = $\overline{WR}$-Ende 1/2 Zyklus früher
D_6 0 = $\overline{RD}$-Ende 1/2 Zyklus früher
D_5 frei
D_4 frei
D_3 0 = $\overline{MREQ}$-Ende 1/2 Zyklus früher
D_2 0 = $\overline{IORQ}$-Ende 1/2 Zyklus früher
D_1 0 } Zykluslänge 0 } Zykluslänge 1 } Zykluslänge
D_0 0 } = 4; 1 } = 3; 0 } = 2

Bei „Übertragung" muß das Zeitablaufbyte zweimal folgen, erst für Tor A, dann für Tor B.

Steuerbyte 2A

D_7 1
D_6 DMA-Freigabe
D_5 Interruptfreigabe
D_4 Vergleichsbyte folgt
D_3 Maskenbyte folgt
D_2 Stop bei Vergleich
D_1 0
D_0 0

Steuerbyte 2B

D_7 1
D_6 } Be- 0 } byte- 0 } block- 1 } burst; 1 } —
D_5 } triebsart: 0 } weise; 1 } weise; 0 } 1 }
D_4 Interruptsteuerbyte folgt
D_3 Adresse B, „High"-Byte folgt
D_2 Adresse B, „Low"-Byte folgt
D_1 0
D_0 1

Interruptsteuerbyte

D_7 frei
D_6 Interrupt vor BUSRQ
D_5 Status verändert Interruptvektor
D_4 Interruptvektor folgt
D_3 Pulszählerbyte folgt

D_2 Pulse werden erzeugt
D_1 Interrupt bei Blockende
D_0 Interrupt bei Gleichheit

Interruptvektor

D_7 V_7
D_6 V_6
D_5 V_5
D_4 V_4
D_3 V_3

D_2	0	bei Interrupt auf RDY;	0	bei Interrupt auf Gleichheit;	1	bei Interrupt auf Block ende;	1	bei Interrupt auf Gleichheit oder Blockende
D_1	0		1		0		1	

D_0 0

Steuerregister 2C

D_7 1
D_6 frei
D_5 0 Stop bei Blockende; 1 wiederholen bei Blockende
D_4 0 $\overline{CE}$ nur; 1 $\overline{CE}/\overline{WAIT}$
D_3 0 RDY aktiv „Low"; 1 RDY aktiv „High"
D_2 frei
D_1 1
D_0 0

Steuerregister 2 D

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	0	0	0	0				Chip zurücksetzen;
1	0	0	0	1				gleicht Zeitverhalten DMA-Tor A an CPU an;
1	0	0	1	0				gleicht Zeitverhalten DMA-Tor B an CPU an;
1	0	0	1	1				0 → Bytezähler, laden Startadressen*);
1	0	1	0	0				Ausführung wird bei momentanem Adreßstand und voller Blocklänge begonnen (Restart);
0	1	0	1	0				Interruptfreigabe;
0	1	0	1	1				Interrupt sperren;
0	1	0	0	0				alle Interrupt werden zurückgesetzt;

0	0	0	0	1	Chipfreigabe, Operationen werden freigegeben;
0	0	0	0	0	Chipsperre, Operationen werden blokkiert;
0	1	1	1	0	Lesemaske folgt;
0	1	0	0	1	das erste Register, das durch Lesemaske frei ist, wird als nächstes gelesen;
0	1	1	1	1	als nächstes Register wird das Statusregister gelesen;
0	1	1	0	0	erzwungenes RDY, es wird intern ein RDY erzeugt;
0	1	1	0	1	vom DMA kommt keine Busanforderung, solange das RETI nicht kommt.
0	0	0	1	0	Rücksetzen Vergleichs-Bit und Blockende-Bit im Statusregister.

*) Bekommt der betreffende Kanal eine Festadresse, so ist das Laden der Startadresse (für Festadresse) nur möglich, wenn dieser Kanal als Eingabekanal definiert ist.

Statusregister

D_7	frei
D_6	frei
D_5	Blockende-Bit, 0 = Blockende
D_4	Vergleichs-Bit, 0 = Vergleich
D_3	0 INT-Pending
D_2	frei
D_1	0 RDY = aktiv
D_0	0 DMA-Zyklus nicht erschienen 1 DMA-Zyklus ist erschienen

Lesen der DMA-Register

Folgende DMA-Register können sequentiell in der aufgezählten Reihenfolge gelesen werden:
Statusregister,
Bytezähler niederwertiger Teil,
Bytezähler höherwertiger Teil,
Adresse A niederwertiger Teil,
Adresse A höherwertiger Teil,
Adresse B niederwertiger Teil,
Adresse B höherwertiger Teil.

Das Weiterschalten von Register zu Register geschieht intern. Soll ein Register nicht gelesen werden, so kann es durch Setzen von „0" an der entsprechenden Stelle der Lesemaske übersprungen werden. Nach RESET und Lesen Statusregister weist der interne Zeiger immer auf das Statusregister.

Lesemaske:

D_7
D_6 Adresse B HWT
D_5 Adresse B NWT
D_4 Adresse A HWT
D_3 Adresse A NWT
D_2 Bytezähler HWT
D_1 Bytezähler NWT
D_0 Statusregister

2.5.4. Beispiel zum DMA-Baustein

Ein DMA-Baustein soll die Ausgabe eines Datenblocks aus dem Speicher auf einen über ein PIO angeschlossenen Drucker steuern. Bild 2.13 zeigt die notwendigen Verbindungen zwischen *U 880*, PIO und DMA. Der Drucker ist am Kanal A des PIO angeschlossen und arbeitet im Hand-shake-Betrieb mit den Signalen ARDY und $\overline{\text{ASTB}}$. Kanal A des PIO wird in die Betriebsart „Ausgabe" initialisiert. Interrupt werden gesperrt. Der DMA wird in die Betriebsart „Byteweise" und in die Funktionsart „Übertragung" initialisiert. Kanal A soll den Speicher, Kanal B das PIO steuern. Solange am PIO das Signal ARDY aktiv ist (Daten sind nicht vom Drucker abgeholt), soll der DMA den Bus freigeben. Dazu wird ARDY mit RDY des DMA verbunden und für RDY aktiv „Low" festgelegt. Durch den Adreßdecoder sollen folgende Adressen festgelegt werden:

PIO	Datenwort	Kanal A	Adresse 00H
	Steuerwort	Kanal A	Adresse 40H
	Datenwort	Kanal B	Adresse 80H
	Steuerwort	Kanal B	Adresse C2H
DMA	Bausteinauswahl		Adresse 30H

Zur Initialisierung der Übertragung vom Speicher zum Drucker müssen folgende Bit-Muster von der CPU *U 880* ausgegeben werden:

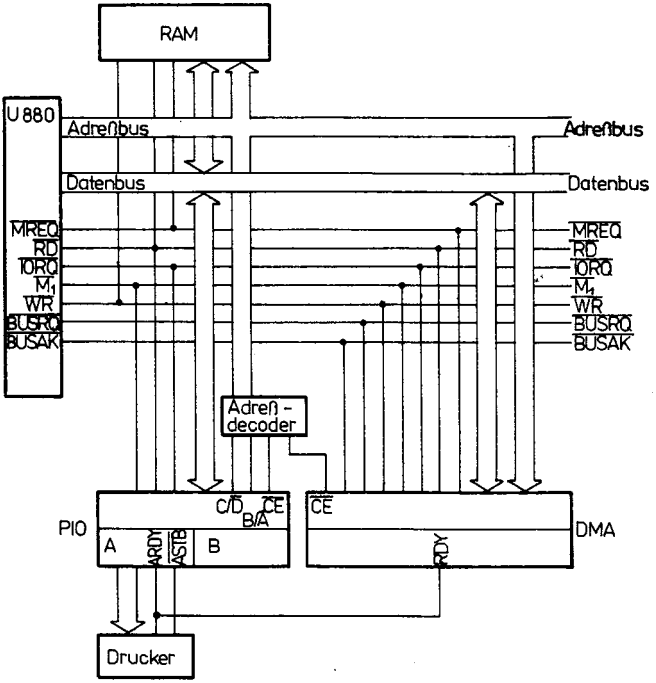


Bild 2.13 Anschluß eines DMA-Bausteins an den Systembus *U 880*

	Befehl	Wort auf Datenbus
Ausgabe Steuerwort zur Festlegung der Betriebsart „Ausgabe" im PIO Kanal A	OUT 40 H	00001111
Interruptsperre für PIO Kanal A	OUT 40 H	00000011
Steuerbyte 1A für BMA Übertragung B → A Da Kanal B mit einer festen Adresse versehen wird, muß er erst als Eingabekanal definiert werden.	OUT 30H	00000001
Steuerbyte 1B für	OUT 30H	00101000

DMA Kanal B		
Kanal B zur Peripherie-		
steuerung mit fester Adresse		
Steuerbyte 2B	OUT 30 H	10000101
Byteweise Übertragung		
Festlegen Adresse für		
Kanal B		
Adresse PIO Datenkanal A	OUT 30 H	PIO-Adresse
zum DMA Kanal B		
Steuerbyte 2D	OUT 30 H	11001111
Laden Toradresse B		
Steuerbyte 1A für DMA	OUT 30 H	01111101
Übertragung A → B		
Kanal A „Low"-Byte	OUT 30 H	„Low"-Byte
Kanal A „High"-Byte	OUT 30 H	„High"-Byte
Blocklänge „Low"-Byte	OUT 30 H	„Low"-Byte
Blocklänge „High"-Byte	OUT 30 H	„High"-Byte
Steuerbyte 1B für	OUT 30 H	00010100
DMA-Kanal A		
Kanal steuert Speicher,		
Adresse wird incrementiert,		
kein Zeitsteuerbyte		
Steuerbyte 2C	OUT 30 H	10000010
RDY aktiv „Low"		
kein WAIT-Status		
Steuerbyte 2D	OUT 30 H	11001111
Laden Toradresse A		
Löschen Blockzähler		
Steuerbyte 2D	OUT 30 H	10000111
Starten DMA-Operation		

3. Programmier- und Anwendungsbeispiele kleiner Mikrorechner

Zur Realisierung der Grundfunktion eines Mikrorechners benötigt man zunächst eine Minimalkonfiguration. Sie ist unbedingt notwendig, um mit dem Mikroprozessor einfache Aufgaben lösen zu können.
Zu dieser Minimalkonfiguration gehören
- der Mikroprozessor mit Stromversorgung, Aussteuerung und Taktgenerator,
- ein Speicher, bestehend aus ROM- und RAM-Bereich,
- ein Anzeigefeld (hexadezimal oder dual),
- eine Eingabetastatur.

Im folgenden Teil sollen einige Beispiele zum Aufbau der Minimalkonfiguration sowie einige Anwendungsbeispiele dieser Konfiguration beschrieben werden.

3.1. Steuerung des Prozessors

Ausgehend vom Prozessor legt man zunächst fest, welche Signale zum Systembus gehören sollen. Dazu zählen die Adreßsignale, die Datensignale und die Steuersignale des Prozessors. Bei Minimalkonfigurationen kann man dabei einige Signale, die nicht benötigt werden, weglassen (z. B. $\overline{\text{RFSH}}$, wenn ohne dynamische Speicher gearbeitet wird, $\overline{\text{BUSRQ}}$ und $\overline{\text{BUSAK}}$, wenn kein DMA-Betrieb vorgesehen ist). Weitere Leitungen können spezielle Funktionen übernehmen, die im Prozessor nicht benötigt werden (z. B. andere Betriebsspannungen, Sperrsignale für den Speicher).
Alle an den Systembus anzuschließenden Funktionseinheiten müssen so aufgebaut sein, daß sie sich mit den Signalen des Systembusses vollständig steuern lassen.
Zur Steuerung des Prozessors benötigt man außer der Betriebsspannung und dem Taktgenerator eine Logik, mit der ein Programm taktweise oder befehlsweise abgearbeitet werden kann. Kommen keine dynamischen Speicher zur Anwendung, so läßt sich das sehr einfach über die $\overline{\text{WAIT}}$-Leitung realisieren (Bild 3.1).

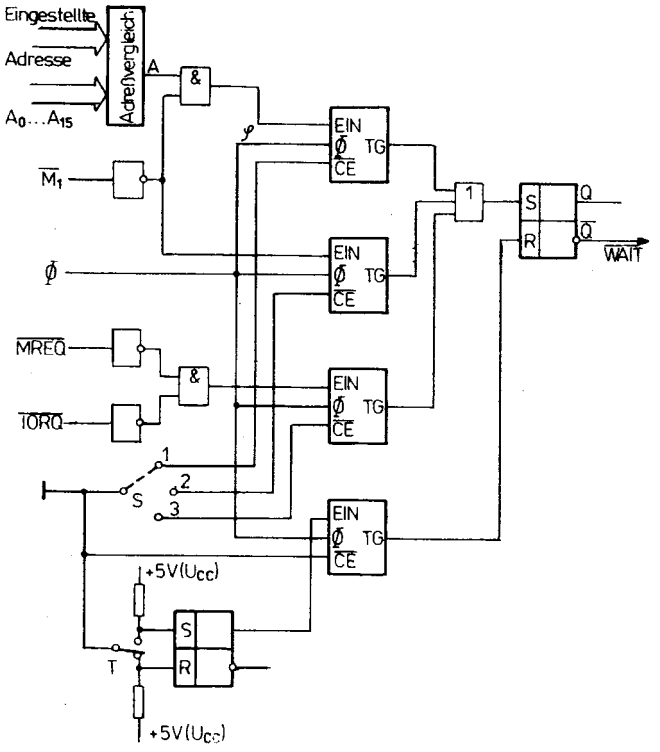


Bild 3.1 Erzeugung des WAIT-Signals zur Realisierung einer taktweisen und befehlsweisen Abarbeitung

Die Schaltung TG soll als Eintaktgenerator arbeiten. Am Ausgang TG wird, wenn $\overline{CE}$ Tiefpegel hat, der erste Takt Φ, nachdem der Eingang EIN Hochpegel erhält, durchgelassen. Die Schaltung „Adreßvergleich" muß so arbeiten, daß am Ausgang A Hochpegel liegt, wenn die eingestellte Adresse mit der an A_0 bis A_{15} liegenden Adresse übereinstimmt. Mit Hilfe des Schalters S kann man wählen, ob das Programm in jedem Zyklus (Stellung S3), nach jedem Befehl (Stellung S2) oder bei einem Befehl mit eingestellter Adresse (Stellung S1) anhält. In Stellung S3 erzeugt die Schaltung mit jedem $\overline{MREQ}$ oder $\overline{IORQ}$ ein $\overline{WAIT}$-Signal, d. h., der Pro-

zessor unterbricht bei jedem Speicherzugriffs- und E/A-Zyklus. In Stellung S2 hält der Prozessor nach dem Erscheinen eines $\overline{M1}$ und in Stellung S1 nach dem Erscheinen eines $\overline{M1}$, wenn gleichzeitig die eingestellte Adresse mit der Busadresse übereinstimmt, an.

3.2. Bedienelemente

Die Bedienelemente dienen zum Anzeigen der Register der CPU, Füllen von Speicherplätzen des RAM und Starten eines Programms. Angezeigt bzw. verändert werden können unmittelbar zunächst nur die Informationen auf dem Systembus. Inhalte von Registern oder Speicherzellen müssen zur Anzeige erst auf den

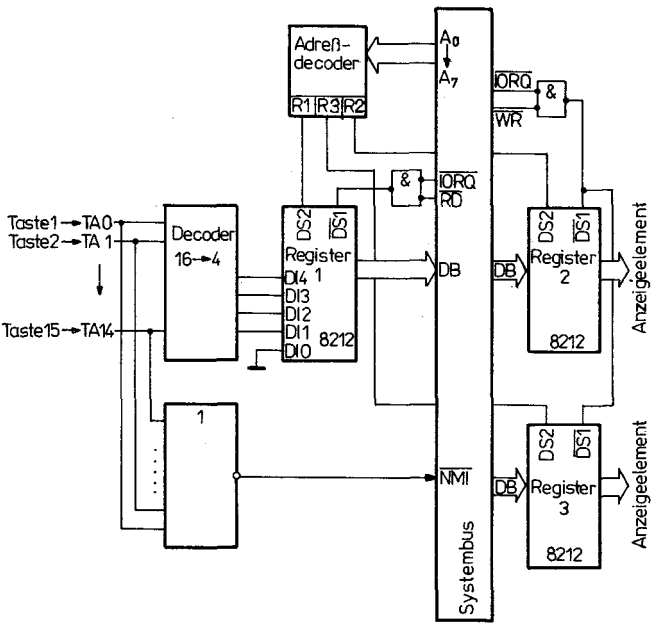


Bild 3.2 Logisches Prinzipschaltbild zur Realisierung von Tastenfunktionen am Mikrorechner (statt TA 14 lies TA 15, statt TA 15 lies TA 16)

Systembus gebracht werden bzw. zur Eingabe vom Systembus geholt werden. Dazu gibt es die Befehle zur Ein- bzw. Ausgabe von Bit-Mustern. Um die oben genannten Funktionen zu realisieren, müssen kleine Programme für diese Funktionen in einem speziellen Speicherbereich (ROM) stehen. Der Zugang zu diesen Programmen ist z. B. über den nichtmaskierten Interrupt möglich.

Beispiel

Jeder Funktion (z. B. Anzeige Register A, Füllen Register A) ist ein Programm zugeordnet, das über eine Taste angesprungen wird. Durch das Drücken der Taste wird ein NMI ausgelöst und gleichzeitig die Nummer der Taste in einem Register gespeichert (z. B. 8212). Mit Hilfe der Nummer läßt sich das zur Taste gehörende Programm finden. Bild 3.2 zeigt das logische Prinzipschaltbild für 16 Tastenfunktionen. Beim Drücken einer Taste entsteht eines der Tastensignale TA0 bis TA15 Am Ausgang des „Decoders 16 → 4" entsteht die Nummer der Taste. Sie wird ins Register 1 gebracht. Gleichzeitig entsteht das Signal $\overline{\text{NMI}}$, das einen Interrupt im Prozessor erzeugt und damit einen Sprung in das Bedienprogramm zum nichtmaskierten Interrupt auslöst. Das Bedienprogramm liest Register 1 und geht damit zum Programm „Anzeige Register A" über. Dieses Programm bringt den Inhalt von Register A zum Register 2 und von da aus zur Anzeige. Je nach Art der Anzeige (dual oder hexadezimal) erfolgt die Auswertung des Inhalts von Register 2. Bei hexadezimaler Anzeige kann der Hexadezimalcode in die beiden Register (Register 2 und Register 3) gebracht werden und von dort über Anzeigeverstärker zu den LED-Elementen.

Die folgende Befehlsliste zeigt den Aufbau des Bedienprogramms und des Programms „Anzeige Register A", wenn der Inhalt von Register 2 dual (z. B. durch Leuchtdioden) angezeigt wird.

Zur Aufstellung des Programms sei festgelegt:

Adresse von Register 1: 20 H
Adresse von Register 2: 21 H
Adresse von Register 3: 22 H

Bedienprogramm NMI

```
PN    AW    ; Bedienprogramm NMI
ORG   66H   ; Startadresse für NMI-Programm
EXAF        ; Retten der alten Registerinhalte in den
```

```
        EXX                 Registern A' bis H'
        IN      20H     ; Tastennummer nach Register A
        LD      E, A
        LD      D, O    ; Tastennummer in Register DE
        LD      HL,     ; Die Anfangsadresse der Tabelle der
                TTAPR     Adressen für die einzelnen Tastenpro-
                          gramme kommt nach HL
        ADD     HL, DE  ; In HL steht die Adresse, in der die
                          Startadresse des Tastenprogramms steht,
                          in unserem Beispiel die Adresse für „An-
                          zeige Register A"
        LD      E, M
        INC     HL
        LD      D, M    ; In DE steht jetzt die Startadresse von
                          „Anzeige Register A"
        EX      DE, HL
        JMP     M       ; Sprung ins Programm „Anzeige Re-
                          gister A"
TTAPR:  DA ATAO; Startadresse für Tastenprogramm Taste 1
        DA ATA1 ; Startadresse für Tastenprogramm Taste 2
          .
          .
          .
        DA  ATA15 ; Startadresse für Tastenprogramm Taste
                    15
Programm „Anzeige Register A"
ATAO: OUT  21H  ; Inhalt Register A nach Register 2
WS:   NOP       ; Warteschleife auf die nächste Tasten-
                  funktion
        JR   WS   — ǂ
        END
```

3.3. Anzeige mit LED-Elementen

Lichtemitteranzeigen mit LED dienen dazu, die im Rechner gespeicherten Zahlen anzuzeigen. Dazu werden in den meisten Fällen 7-Segment-Elemente verwendet (s. Teil 1, Abschnitt 3.5.). Bild 3.3 zeigt den Anschluß von 6 LED-Anzeigen an den Systembus eines Mikrorechners. Der Systembus erhält die Bussignale

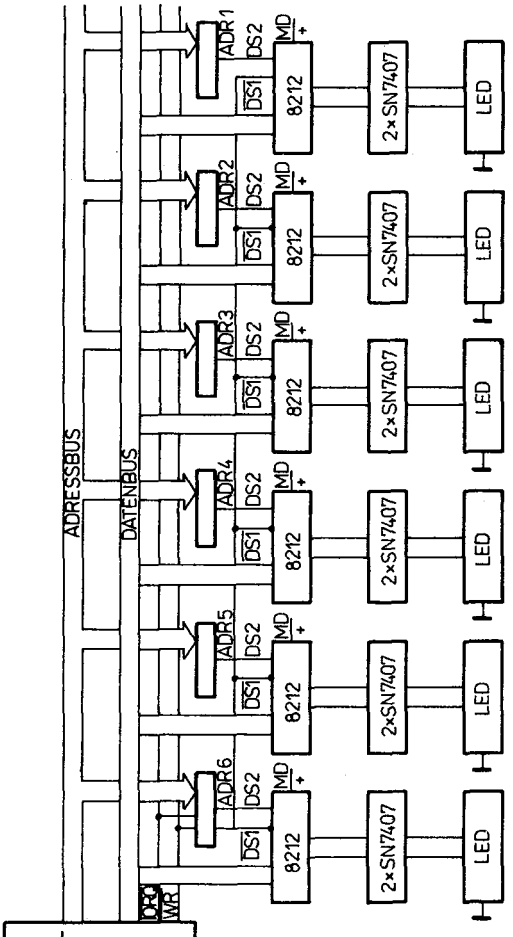


Bild 3.3 Anschluß von 6 LED-Anzeigen an den Systembus eines Mikrorechners

eines Prozessors (Adreß-, Daten- und Steuerbus). Zu den einzelnen Anzeigen führen Ausgabetore, über die die Information vom Datenbus zur LED-Anzeige gelangt.

Die Ausgabe läßt sich im einfachsten Fall über eine Torschaltung

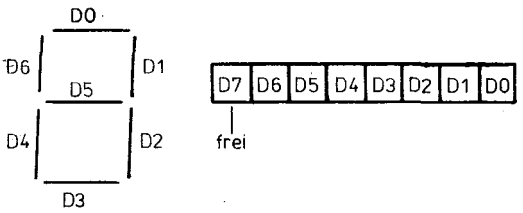


Bild 3.4 Zuordnung der Bit-Stellen eines Bytes zu den Segmenten einer LED-Anzeige

realisieren. Besser ist es jedoch, wenn das zur LED-Anzeige erforderliche Bit-Muster in einem Register zwischengespeichert wird. Dazu eignet sich z. B. der Baustein *8212* (s. Teil 1, Abschnitt 3.). Für die Anzeige ist es nicht notwendig, daß die ausgegebenen Ziffern über einen BCD-zu-7-Segment-Decoder umgewandelt werden. Statt dessen werden die zur Aussteuerung eines Anzeigeelements notwendigen Bit-Muster vom Prozessor direkt ausgegeben. Die Umwandlung vom BCD-Code in das notwendige Bit-Muster läßt sich dann durch ein Programm erreichen. Bild 3.4 zeigt, wie den einzelnen Segmenten die Bit eines Bytes zugeordnet werden können. Daraus ergibt sich folgende Bit-Muster-Tabelle für die Hexadezimalziffern 0 bis F:

Zeichen	Bit-Muster (dual)								hexadezimal
0	0	1	0	1	1	1	1	1	5 F
1	0	0	0	0	0	1	1	0	0 6
2	0	0	1	1	1	0	1	1	3 B
3	0	0	1	0	1	1	1	0	2 E
4	0	1	1	0	0	1	1	0	6 6
5	0	1	1	0	1	1	0	1	6 D
6	0	1	1	1	1	1	0	1	7 D
7	0	0	0	0	0	1	1	1	0 7
8	0	1	1	1	1	1	1	1	7 F
9	0	1	1	0	1	1	1	1	6 F
A	0	1	1	1	0	1	1	1	7 7
B	0	1	1	1	1	1	0	0	7 C
C	0	1	0	1	1	0	0	1	5 9
D	0	0	1	1	1	1	1	0	3 E
E	0	1	1	1	1	0	0	1	7 9
F	0	1	1	1	0	0	0	1	7 1

Im Beispiel sollen die Tore zu den einzelnen Anzeigeelementen folgende Adressen haben:

Anzeigenelement	Adresse
1	0 CH
2	0 DH
3	0 EH
4	0 FH
5	1 CH
6	1 DH

Die Adressen der Anzeigeelemente sind in einer Tabelle mit der Anfangsadresse TAB gespeichert. TAB ist eine symbolische Adresse, die beim Abspeichern der Toradressen durch eine beliebige Speicheradresse ersetzt werden kann.

Die zur Anzeige kommenden 6 Ziffern sind in einer Tabelle mit der Anfangsadresse DAT zusammengefaßt. Die Ziffer steht dabei in den letzten 4 Bit der genannten Zelle in dualer Darstellung:

	7	6	5	4	3	2	1	0
Zelle DAT	0	0	0	0	Ziffer 1			
Zelle DAT + 1	0	0	0	0	Ziffer 2			
Zelle DAT + 2	0	0	0	0	Ziffer 3			
Zelle DAT + 3	0	0	0	0	Ziffer 4			
Zelle DAT + 4	0	0	0	0	Ziffer 5			
Zelle DAT + 5	0	0	0	0	Ziffer 6			

Beschreibung des Programms zur Realisierung der Anzeige

Die 1. Ziffer wird aus der Tabelle DAT entnommen und das zu dieser Ziffer gehörige Bit-Muster aus der Bit-Muster-Tabelle für die Hexadezimalziffern herausgesucht. Dieses Bit-Muster wird anschließend über Tor 1 ausgegeben.

Danach wird das gleiche für die Tore 2 bis 6 durchgeführt.

Bild 3.5 zeigt das Flußdiagramm.

Das Programm zur Realisierung der Anzeige lautet:

```
001                         PN LEDANZEIGE/PROGRAMM-
                            ANZEIGE VON 6 ZIFFERN AUF
                            LEDELEMENT
002  000 000 006 07         LEDA: LD B, 6  /FUER 6 ZIFFERN
     000 001 006
003  000 002 335 14         LD IX, 3C 30H  /ADRESSE
                                           DATENPUFFER
     000 003 041
```

```
                000 004 060
                000 005 074
004             000 006 375 14      LD IY, 1B 15H   /ADRESSE DER
                                                    TORADRESS-
                                                    TABELLE
                000 007 041
                000 010 325
                000 011 003
005             000 012 041 10      LD HL, 1B 1BH   /ADRESSE DER
                                                    SEGMENT-
                                                    CODETABELLE
                000 013 333
                000 014 003
006             000 015 375 19      ZYK: LD C, (IY) /TORADRESSE
                                                    INS C-RE
                                                    /GISTER
                000 016 116
                000 017 000
007             000 020 335 19      LD E, (IX)      /ZIFFER NACH E
                000 021 136
                000 022 000
008             000 023 026 07      LD D, 0
                000 024 000
009             000 025 031 11      ADD HL, DE      /IN HL STEHT
                                                    ADRESSE DES
                                                    SEGMENTCODES
010             000 026 176 07      LD A, M-        /SEGMENTCODE
                                                    NACH A
                                                    /REGISTER
011             000 027 355 12      OUT A           /ZIFFER
                                                    AUSGEBEN
                000 030 171
012             000 031 335 09      INC IX          /DATENPUFFER
                                                    ADRESSE
                                                    WEITER-
                                                    SCHALTEN
```

```
      000 032 043
013   000 033 375 09      INC IY          /TORADRESS-
                                          TABELLEN-
                                          ADRESSE WEI-
                                          TERSCHALTEN
      000 034 043
014   000 035 020 13      DJNZ ZYK- #
      000 036 356         RET
015   000 037 311 10      END             GESAMTTAKT-
                                          ANZAHL = 000161
```

Im Programm bedeuten die einzelnen Spalten:
Spalte 1 enthält die laufende Nummer der Befehlszeile.
Spalte 2 enthält die Adresse der Speicherzelle, in der der Befehl

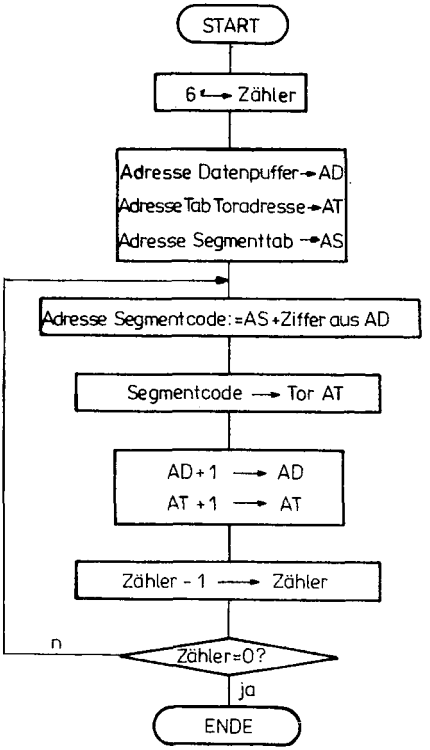


Bild 3.5 Flußdiagramm für die Ausgabe von 6 BCD-Ziffern auf LED-Anzeigen

gespeichert ist. Dabei ist eine Darstellung gewählt worden, in der das höherwertige und das niederwertige Byte der Adresse in oktaler Schreibweise dargestellt ist.
Spalte 3 enthält den Inhalt der Speicherzelle in oktaler Form. In Spalte 3 ist somit das Maschinencodeprogramm aufgeführt, d. h. die codierte Darstellung der einzelnen Befehle.
Spalte 4 enthält die Anzahl der Takte, die für die Ausführung des betreffenden Befehls benötigt wird.
Spalte 5 enthält das Programm in Assemblerschreibweise mit den Spalten Marke, Bezeichnung der Operation, Bezeichnung des Operanden, Bemerkung.
Die Schreibweise ZYK- # wird dann verwendet, wenn es sich um relative Adressen handelt. Der Assembler (Übersetzer des geschriebenen Programms in die Maschinencodierung) bildet aus ZYK- # den Abstand zwischen dem Befehl ZYK: LD C, (IY) und dem Befehl DJNZ ZYK- # und trägt ihn als relative Adresse in den Befehl DJNZ ZYK- # ein.

3.4. Digitaluhr

Die Zeitfunktion läßt sich auf unterschiedlichste Weise mit Hilfe der Mikrorechnerschaltkreise realisieren. Dabei kann der Takt Φ der Zeitgeber sein. Im einfachsten Fall werden die Zeiteinheiten (Sekunde, Minute, Stunde) mit dem Prozessor erstellt. Dabei benötigt man neben dem Prozessor nur einen kleinen Speicher und eine Anzeige. Soll der Prozessor gleichzeitig mit der Uhrenfunktion noch andere Aufgaben lösen, so ist zusätzlich ein CTC-Schaltkreis erforderlich.

3.4.1. Digitaluhr mit Prozessor und einer LED-Anzeige

Zur Realisierung einer Digitaluhr sei die in Abschnitt 3.3. beschriebene LED-Anzeige verwendet. Die Zuordnung der LED-Elemente zur Uhrzeit zeigt die folgende Darstellung:

LED-Element	6	5	4	3	2	1
	Stunden		Minuten		Sekunden	

Das Programm besteht aus folgenden Teilschritten:

1. Schritt Realisierung des eigentlichen Uhrenprogramms;
2. Schritt Speichern der Zahlenwerte für Sekunden, Minuten und Stunden in den Datenpuffer für die Anzeige;
3. Schritt Aufruf des Programms für die LED-Anzeige.

Für den 1. Schritt läßt sich das in Bild 3.6 gezeigte Ablaufdiagramm aufstellen. Im 2. Schritt werden die Zahlenwerte für Sekunden, Minuten und Stunden der Reihe nach aus den zugehö-

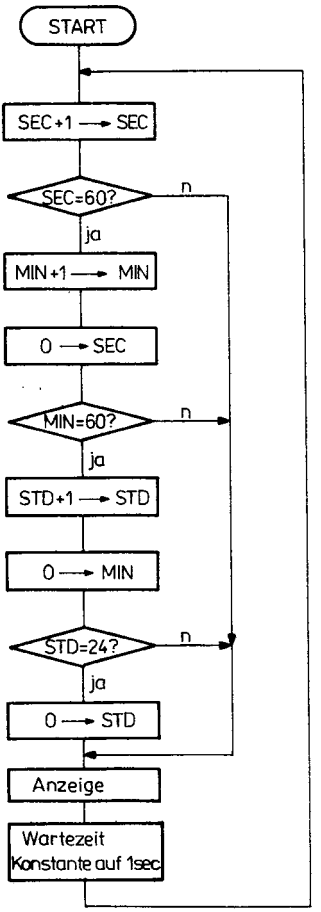


Bild 3.6
Flußdiagramm zur Errechnung von Sekunden, Minuten und Stunden eines Uhrenprogramms

rigen Registern geholt. Nach dem Abtrennen der unteren und oberen 4 Bit speichert man sie der Reihe nach im Datenpuffer für das LED-Anzeigeprogramm.

```
001                          /PROGR. ZUR REALISIERUNG
002                          /DER UHR OHNE
003                          /CTC-SCHALTKREIS
                             /PROGRAMM UHRENFUNKTION
004  000 000 072 13          UHR: LD A, (U) ;Sekundenzählung
     000 001 065
     000 002 000
005  000 003 306 07          ADD 1          /SEK=SEK+1
     000 004 001
006  000 005 047 04          DAA
007  000 006 062 13          LD (U), A
     000 007 065
     000 010 000
008  000 011 326 07          SUB 60H        /60SEK ?
     000 012 140
009  000 013 040 12          JRNZ ANZP- #   /Sprung ins
     000 014 043                            Anzeigeprogramm
010  000 015 062 13          LD (U), A      /Rücksetzen
     000 016 065                            Sekunden
     000 017 000
011  000 020 072 13          LD A, (U + 1)  /Minutenzählung
     000 021 066
     000 022 000
012  000 023 306 07          ADD 1          /MIN=MIN+1
     000 024 001
013  000 025 047 04          DAA
014  000 026 062 13          LD (U + 1), A
     000 027 066
     000 030 000
015  000 031 326 07          SUB 60H        /60MIN ?
     000 032 140
016  000 033 040 12          JRNZ ANZP- #   /Sprung ins
     000 034 023                            Anzeigeprogramm
017  000 035 062 13          LD (U + 1), A  Rücksetzen Minuten
     000 036 066
     000 037 000
018  000 040 072 13          LD A, (U + 2)  /Stundenzählung
```

```
      000 041 067
      000 042 000
019   000 043 306 07    ADD 1              /STD+1
      000 044 001
020   000 045 047 04    DAA
021   000 046 062 13    LD (U + 2), A
      000 047 067
      000 050 000
022   000 051 326 07    SUB 24H            /24 STD ?
      000 052 044
023   000 053 040 12    JRNZ ANZP- #       /Sprung ins
      000 054 003                          Anzeigeprogramm
024   000 055 062 13    LD (U + 2), A      /Rücksetzen
      000 056 067                          Stunden
      000 057 000
025   000 060 315 17 ANZP: CALL ANZE       /Aufruf des
      000 061 072                          Anzeigeprogramms
      000 062 000
026   000 063 030 12    JR UHR- #
      000 064 313
027   000 065 000 00 U: DB 0               /SEK-Speicherzelle
028   000 066 000 00    DB 0               /MIN-Speicherzelle
029   000 067 000 00    DB 0               /STD-Speicherzelle
030   000 070 000 00 XX:DA 0               /Zeitkonstante
      000 071 000       /Anzeigeprogramm
031   000 072 041 10 ANZE: LD HL, LP       /Abspeichern der
      000 073 155                          /Zahlenwerte für
      000 074 000                          /SEK, MIN und
032   000 075 072 13    LD A, (U)          /STD in den LED-
      000 076 065                          /Puffer-LP
      000 077 000
033   000 100 315 17    CALL ABSP          /SEK→LP
      000 101 136
      000 102 000
034   000 103 072 13    LD A, (U + 1)
      000 104 066
      000 105 000
035   000 106 315 17    CALL ABSP          /MIN→LP
      000 107 136
      000 110 000
```

```
036  000 111 072 13      LD A, (U+2)
     000 112 067
     000 113 000
037  000 114 315 17      CALL ABSP       /STD→LP
     000 115 136
     000 116 000
038  000 117 001 10      LD BC, (XX)     /Zeitkonstante
     000 120 070
     000 121 000
039  000 122 305 11 WZYK: PUSH BC        /Zeitschleife für 1 s
040
041  000 123 315 17      CALL LEDA       /Ausgabe der 6
                                         Ziffern in die LED-
                                         Register
     000 124 163
     000 125 000
042  000 126 301 10      POP BC
043  000 127 013 06      DEC BC          /Zählung für 1 s
044  000 130 257 04      XOR A
045  000 131 260 04      OR B
046  000 132 261 04      OR C
047  000 133 040 12      JRNZ WZYK-#     /BC=0?
     000 134 365
048  000 135 311 10      RET
049  /Unterprogramm zum Trennen der beiden BCD-
050  /Ziffern eines 8-Bit-Musters im A-Register
051  /und Abspeichern in den LED-Puffer.
052  /Die Adresse des LED-Puffers steht in HL
053  000 136 107 04 ABSP: LD B, A
054  000 137 346 07      AND 0FH
     000 140 017
055  000 141 167 07      LD M, A         /Speichern untere
056  000 142 043 06      INC HL          4 Bit→LP
057  000 143 170 04      LD A, B
058  000 144 017 04      RRCA
059  000 145 017 04      RRCA
060  000 146 017 04      RRCA
061  000 147 017 04      RRCA
062  000 150 346 07      AND 0FH
     000 151 017
```

```
063  000 152 167 07        LD M, A          /Speichern obere 4
064  000 153 043 06        INC HL           Bit→LP
065  000 154 311 10        RET
066  000 155 000 00 LP:    BER 6            LED-PUFFER
     000 156 000
     000 157 000 00
     000 160 000
     000 161 000 00
     000 162 000
069  /Programm LED-Anzeige
070  /Ausgabe der 6 Ziffern in die LED-Register
071  000 163 006 07 LEDA:  LD B, 6          /FUER 6 ZIFFERN
     000 164 006
072  000 165 335 14        LD IX, LP        /ADRESSE
     000 166 041                            /LEDPUFFER
     000 167 155
     000 170 000
073  000 171 375 14        LD IY, 1B 15H    /ADRESSE DER
     000 172 041                            /TORADRESS-
     000 173 325                            /TABELLE
     000 174 003
074  000 175 041 10        LD HL, 1B 1BH    /ADRESSE DER
     000 176 333                            /SEGMENTCODE-
     000 177 003                            /TABELLE
075  000 200 375 19 ZYK:   LD C, (IY)       /TORADR. INS
     000 201 116                            C-REG.
     000 202 000
076  000 203 335 19        LD E, (IX)       /ZIFFER NACH  E
     000 204 136
     000 205 000
077  000 206 026 07        LD D, 0
     000 207 000
078  000 210 031 11        ADD HL, DE       /IM HL STEHT
                                            /ADRESSE DES
                                            /SEGMENT-
                                            /CODES
079  000 211 176 07        LD A, M          /SEGMENTC.N.A-
                                            REG.
080  000 212 355 12        OUT A            /ZIFFER
     000 213 171                            /AUSGEBEN
```

```
081  000 214 335 09      INC IX          /DATENPUFFER-
                                         /ADRESSE
     000 215 043                         /WEITER-
                                         /STELLEN
082  000 216 375 09      INC IY          /TORADR.-
                                         TABELLE
     000 217 043                         /WEITER-
                                         /STELLEN
083  000 220 020 13      DJNZ ZYK- #
     000 221 356
084  000 222 311 10      RET
```

GESAMTTAKTANZAHL = 000 659

Zur Übersichtlichkeit ist das Gesamtprogramm in einzelne Funktionen unterteilt worden, die als getrennte Unterprogramme ausgeführt sind. Das Gesamtprogramm besteht aus folgenden Programmen bzw. Unterprogrammen:

a) Programm zur Realisierung der Uhrenfunktion;
b) Anzeigeprogramm zum Füllen des Datenpuffers für die LED-Anzeige (LED-Puffer LP) und zur Ausgabe der Werte für Sekunde, Minute und Stunde in die Register der LED-Elemente;
c) Teilprogramm zum Trennen der beiden BCD-Ziffern für Sekunde, Minute und Stunde;
d) Unterprogramm zur Ausgabe der 6 Ziffern für Sekunde, Minute und Stunde in die LED-Register.

Das Programm „Uhrenfunktion" (UHR) realisiert die Zählung von Sekunden, Minuten und Stunden. Von diesem Programm aus erfolgt der Sprung ins „Anzeigeprogramm".
Das Anzeigeprogramm (ANZE) holt den Wert für Sekunden aus der Zelle U und ruft das Programm ABSP auf. ABSP speichert diesen Wert in den LED-Puffer. Danach holt es die Werte für Minuten und Stunden aus den Zellen U + 1 und U + 2 und speichert diese Werte durch Aufrufen von ABSP ebenfalls in den LED-Puffer. Zum Schluß dieses Programms wird das Programm LEDA (Füllen der Register für die LED-Anzeige) aufgerufen. Das Programm LEDA wird so oft durchlaufen, bis genau eine Sekunde vergangen ist. Zu diesem Zweck wird in die Zelle XX eine Zahl (Zeitkonstante) eingetragen, die auf 0 rückgezählt wird. Die Zeitkonstante kann durch Eingabe über die Tastatur so lange

korrigiert werden, bis die Uhr genau geht. Mit ihr läßt sich die Genauigkeit einstellen.
Das Programm zur Trennung der beiden BCD-Ziffern in Sekunde, Minute und Stunde ABSP geht davon aus, daß der Zahlenwert im A-Register steht und die Adresse des LED-Puffers in HL. Es trennt die beiden Ziffern und speichert sie einzeln in den LED-Puffer. Das Programm zur Ausgabe der 6 Ziffern für Sekunden, Minuten und Stunden in die LED-Register (LEDA) entspricht dem in Abschnitt 3.3. aufgestellten Programm.

3.4.2. Digitaluhr mit CTC-Baustein

Der Vorteil dieser Konfiguration ist es, daß das Uhrenprogramm den Rechner nur dann beansprucht, wenn eine Weiterzählung der Zeit (Sekunden) vorgesehen ist. Der Rechner kann damit gleichzeitig für weitere Aufgaben genutzt werden.
Bild 3.7 zeigt die Zusammenstellung der Baugruppen zum Aufbau einer Digitaluhr mit CTC-Baustein.
Das Programm besteht aus folgenden Teilabschnitten:

1. Initialisieren des CTC-Bausteins,
2. Uhrenprogramm,
3. Abspeichern der Zahlenwerte für Sekunden, Minuten und Stunden in den Datenpuffer für die Anzeige (LED-Puffer),
4. Aufruf des Programms für die LED-Anzeige.

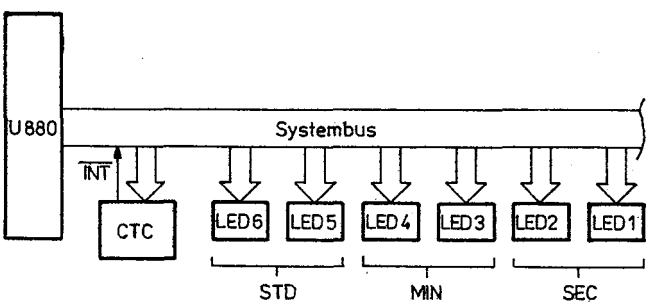


Bild 3.7 Zusammenschaltung der Bausteine *U 880* und *U 880* — CTC zur Realisierung einer Digitaluhr

Der CTC-Baustein zählt die Takte Φ für die Zeiteinheit „1 s". Dazu setzt man Kanal 0 des Bausteins in die Betriebsart „Zeitgeber". Wenn man den Vorteiler nutzt, so können mit einem Kanal $256 \times 256 = 65536$ Takte gezählt werden. Beträgt die Quarzfrequenz z. B. $f = 2{,}4$ MHz, so besteht 1 s aus $2{,}4 \cdot 10^6$ Takten. Ein Kanal reicht daher für die Zählung einer Sekunde nicht aus. Man zählt deshalb im Kanal 0 die Hundertstelsekunden und über Kanal 1 die Sekunden. Der ZC/TOO-Impuls wird dabei für die Zählung im Kanal 1 benutzt.
Wird der Vorteiler von Kanal 0 auf $p = 256$ gestellt, so gilt nach der in Abschnitt 2.2. angegebenen Formel für die Zeitdifferenz der Nulldurchgänge:

$$t_1 = t_c \cdot p \cdot TKO = \frac{p \cdot TKO}{f} \cdot$$

Mit $t_1 = 0{,}01$ s, $p = 256$, $f = 2{,}4576$ MHz $= 2{,}4576 \cdot 10^6\ s^{-1}$ gilt:

$$0{,}01\ s = \frac{256 \cdot TKO}{2{,}4576.10^6 \cdot s^{-1}} \cdot$$

Daraus folgt:

$$TKO = \frac{24576}{256} = 96.$$

Im Kanal 0 des CTC-Bausteins wird demzufolge die Zeitkonstante $TKO = 96$ eingestellt.
Kanal 1 wird in die Betriebsart „Zähler" gesetzt. Die Zeitkonstante beträgt $TK1 = 100$. Damit durchläuft Kanal 1 jede Sekunde den Zählerstand „0". Mit diesem Nulldurchgang bildet man eine Interruptanforderung an den Prozessor. Für die Initialisierung des CTC-Bausteins werden für das Beispiel folgende Adressen vereinbart:

1. Die Auswahladresse des CTC-Bausteins sei 80 H.
2. Der Interruptvektor des CTC-Bausteins sei 90 H.
3. Die Taktfrequenz des Rechners sei 2,4576 MHz.

```
; Uhrenprogramm mit CTC-Baustein
        PN   UHRCTD
; Initialisierung CTC
UCTC:   LD   A,O              ; Höherwertiger Teil Interruptvektor
        LD   I,A
        LD   A,90H            ; Niederwertiger Teil Interruptvektor
        OUT  80H
        LD   IX, ABED         ; Adresse Bedienprogramm CTC-
                              Interrupt
        LD   (90H), IX        ; für Kanal 1
        LD   A, 37H           ; Kanalsteuerwort für Kanal 0
                              ; Betriebsart Zeitgeber, Interrupt-
                                verbot,
                              ; Vorteiler 256, Zeitkonstante laden
        OUT  80H
        LD   A, (TKO)         ; Zeitkonstante für Kanal 0
        OUT  80H
        LD   A, OC7H          ; Kanalsteuerwort für Kanal 1
                              ; Betriebsart Zähler, Interrupt
                              ; erlaubt, Zeitkonstante laden
        OUT  81H
        LD   A, (TK1)         ; Zeitkonstante für Kanal 1
        OUT  81 H
        IM   2                ; Interruptmode 2
        EI                    ; Interruptfreigabe
; Anzeigeprogramm
ANZ:    LD HL, LEDB           ; Adresse LED-Puffer nach HL
        LD A, (UHR)           ; Sekunden nach LED-Puffer
        CALL ABSP
        LD A, (UHR+1)         ; Minuten nach LED-Puffer
        CALL ABSP
        LD A, (UHR+2)         ; Stunden nach LED-Puffer
        CALL ABSP
        CALL LED              ; SEC, MIN und STD in
                              ; Anzeigeregister
        JR ANZ — #
; Interruptbedienprogramm (Zeitzählprogramm)
ABED:   PUSH HL
        PUSH AF
SEC:    LD HL, UHR            ; Sekundenzähleradresse nach HL
        LD A, (HL)
        ADD 1                 ; SEC+1 → SEC
        DAA
        LD (HL), A
        CMP 60H               ; SEC = 60 ?
        JRZ MIN — #
        JR EN — #
```

```
MIN:    LD (HL), 0          ; 0 nach SEC
        INC HL              ; Minutenzähleradresse nach HL
        LD A, (HL)
        ADD 1               ; MIN + 1 → MIN
        DAA
        LD (HL), A
        CMP 60H             ; MIN = 60 ?
        JRZ STD — #
        JR EN — #
STD:    LD (HL), 0          ; 0 nach MIN
        INC HL              ; STD-Zähler nach HL
        LD A, (HL)
        ADD 1
        DAA                 ; STD + 1 → STD
        LD (HL), A
        CMP 24H             ; STD = 24 ?
        JRZ RES — #
        JR EN — #
RES:    LD (HL), 0          ; 0 nach STD
EN:     POP AF
        POP HL
        EI                  ; Interruptfreigabe
        RET
; Abspeicherprogramm für 2 BCD-Ziffern
; aus Register A in einem LED-Puffer
; dessen Adresse in HL steht.
ABSP:   PUSH AF
        AND OFH
        LD (HL), A          ; Untere 4 Bit nach LED-Puffer
        INC HL
        POP AF
        RRCA
        RRCA
        RRCA
        RRCA
        AND OFH
        LD (HL), A          ; Obere 4 Bit nach LED-Puffer
        INC HL
        RET

; Ausgabeprogramm der Ziffern vom LED-Puffer
; in die Anzeigeregister
LED:    LD B,6              ; Zähler für 6 Ziffern
        LD IX, LEDB         ; Adresse LED-Puffer nach IX
        LD IY, ANR          ; Adresse Anzeigeregistertabelle
                            ; nach IY
AUS:    LD C, (IY)          ; Adresse Anzeigeregister nach C
        LD HL, SEGM         ; Adresse der Segmenttabelle nach HL
        LD E, (IX)          ; Ziffer nach DE
        LD D, 0
```

```
        ADD HL, DE          ; In HL steht Adresse des Segment-
                            ; codes der Ziffer
        LD   A, (HL)
        OUT  A              ; Ziffer ausgeben
        INC  IX
        INC  IY
        DJNZ AUS — #        ; nächste Ziffer
        RET
LEDB:   BER  6              ; 6 Zellen für LED-Puffer
ANR:    DB   OCH            ; Tabelle der Adressen der Anzeigereg.
        DB   ODH
        DB   OEH
        DB   OFH
        DB   1 CH
        DB   1DH
SEGM:   DB   05FH           ; Segmenttabelle
        DB   6H
        DB   3BH
        DB   2EH
        DB   66H
        DB   6DH
        DB   7DH
        DB   7H
        DB   7FH
        DB   6FH
UHR:    BER  3              ; Speicher für SEC-, MIN-, STD-
                            ; Werte
TKO:    DB   60H            ; Zeitkonstante für CTC-Kanal 0
TK1:    DB   64H            ; Zeitkonstante für CTC-Kanal 1
        END
```

Ausgabe der Information aus dem Speicher auf eine LED-Anzeige

Ähnlich wie das Uhrenprogramm kann man die Ausgabe der Information aus dem Speicher auf eine LED-Anzeige realisieren. Da auf der LED-Anzeige immer nur der Inhalt einer Speicherzelle anzeigbar ist, schreibt man ein Ausgabeprogramm so, daß nach einer vorgegebenen Zeit Δt der Inhalt der nächsten Speicherzelle angezeigt wird. Auf diese Art und Weise wird der Inhalt des gewünschten Speicherbereiches nacheinander angezeigt.
Mit 6 LED-Elementen kann man die Adresse und den Inhalt der betreffenden Speicherzelle gleichzeitig anzeigen. Für das folgende Programm gilt folgende Zuordnung:

LED 6	LED 5	LED 4	LED 3	LED 2	LED 1
Adresse	Adresse	Adresse	Adresse	Daten	Daten
A15–A12	A11–A8	A7–A4	A3–A0	D7–D4	D3–D0

Das folgende Programm zeigt die Anzeige eines Speicherbereiches, dessen Anfangsadresse in HL und dessen Blocklänge in BC steht, auf einer 6stelligen LED-Anzeige.
Nach einer gewählten Zeit TK wird jeweils der nächste Wert angezeigt. Am Schluß erscheint das Abschlußbild.

Das Programm besteht aus folgenden Teilen:
- Übertragung des Inhaltes und der Adresse der gewählten Speicherzelle in einen Datenpuffer DP
- Trennen der 6 Ziffern des Datenpuffers und Übertragung in den LED-Puffer LP
- Anzeige aus dem LED-Puffer LP
- Zeitprogramm mittels CTC

Die Adressen für CTC und Interruptvektor sind die gleichen wie beim Uhrenprogramm.

```
        PN SPAUS
; Anzeigeprogramm eines Speicherbereiches über LED-Elemente
; Vor dem Start steht die Anfangsadresse des Speicherbereiches in
; HL und die Blocklänge in BC
; Auf 4 LED-Elementen erscheint die Adresse
; Auf 2 LED-Elementen der Inhalt der Speicherzelle
; Nach einer Zeit TK wird jeweils der nächste Wert angezeigt

; CTC-Initialisierungsprogramm
INIT:   LD   A, 0           ; Höherwertiger Teil des Interrupt-
                            ; vektor
        LD   I, A
        LD   A, 90H         ; Niederwertiger Teil des Interrupt-
                            ; vektor
        OUT     80H
        LD   IX, ABED       ; Adresse Auswahlprogramm für den
                            ; Anzeigewert
        LD   (90H), IX
        LD   A,37H          ; Kanalsteuerwort für Kanal 0
                            ; Betriebsart-Zeitgeber
        OUT     80H
        LD A, (TKO)         ; Zeitkonstante für Kanal 0
        OUT     80H
        LD A, OC7H          ; Kanalsteuerwort für Kanal 1
                            ; Betriebsart-Zähler, Interrupt
                            ; erlaubt
        OUT     81H
        LD A, (TK1)         ; Zeitkonstante für Kanal 1
        OUT     81
        IM 2
MO:     EI                  ; Warteschleife auf Interrupt
        JR      MO — #

            ; Auswahlprogramm für den anzuzeigenden Wert
ABED:   LD A, (HL)          ; Inhalt der anzuzeigenden Speicher-
                            ; zelle nach A
        LD (DP), A
        LD (DP + 1), HL     ; Adresse der anzuzeigenden Speicher-
                            zelle nach DP + 1 und DP + 2 ;
        PUSH    HL          ; Retten von HL und BC
        PUSH    BC
        CALL    DPLP        ; Überführung der Anzeigewerte vom
                            ; Datenpuffer in den LED-Puffer
        CALL    ANZ         ; Anzeige der Werte aus dem LED-
                            ; Puffer. Die Code für die LED-
                            ; Elemente werden zwischenge-
                            ; speichert
        POP     BC
        POP     HL
```

```
        DEC     BC
        XOR     A
        OR      B
        OR      C
        JRNZ    M1 — #      ; Test auf Ende
        CALL    ENDE
M1:     RET                 ; Rücksprung in die Warteschleife
                ; Programm zur Überführung der Werte vom Daten-
                ; puffer
                ; zum LED-Puffer mit getrennter Abspeicherung der
                ; Hexadezimalstellen
DPLP:   LD      B, 3        ; Für 3 Byte
        XOR     A           ; Löschen Register A
        LD      HL, DP
        LD      DE,LP
ZYK:    RRD                 ; Niederwertige Hexadezimalstelle
                            ; von DP nach LED-Puffer
        LD      (DE), A
        RRD                 ; Höherwertige Hexadezimalstelle
                            ; von DP nach LED-Puffer
        LD      (DE),A
        INC     DE
        INC     HL
        DJNZ    ZYK — #     ; Wiederholung bis alle Zellen des
                            ; DP abgearbeitet
        RET
                ; Programm zur Anzeige der Hexadezimalziffern
                ; aus dem LED-Puffer
ANZ:    LD      B, 6        ; Für 6 Ziffern
        LD      IX,LP       ; Adresse LED-Puffer
        LD      IY,TOR      ; Adresse der Tabelle der
                            ; peripheren LED-Adressen
ZYO:    LD      HL,STAB     ; Adresse der Segmenttabelle
        LD      C, (IY)     ; periphere LED-Adresse nach
                            ; Register C
        LD      E, (IX)     ; Hexadezimalziffer nach Register E
        LD      D, 0
        ADD     HL, DE      ; In HL steht Adresse des
                            ; Segmentcodes
        LD      A, (HL)     ; Segmentcode nach A
        OUT     A           ; Ziffer ausgeben
        INC     IX          ; LED-Pufferadresse weiterschalten
        INC     IY          ; Weiterschalten Adresse für
                            ; LED-Adreßtabelle
        DJNZ ZYO — #        ; Wiederholung bis alle 6 Ziffern
                            ; ausgegeben
        RET
```

```
                              ; Abschlußprogramm
            ; CTC-Zählung anhalten
ENDE:  LD    A,3              ; Kanalsteuerwort für CTC anhalten
       OUT   80H
       OUT   81H
            ; Ausgabe des Abschlußstriches
       LD    B,6              ; Für 6 LED-Elemente
       LD    IY, TOR          ; Adresse der Tabelle der LED-
                              Adressen
       LD    A,8              ; Segmentcode für Abschlußstrich
ZY1:   LD     C, (IY)         ; periphere LED-Adresse nach
                              ; Register C
       OUT   A
       IN C  IY
       DJNZ  ZY1 — #
       HALT

            ; Datenbereiche
LP:    BER    6               ; 6 Zellen für LED-Puffer
DP:    BER    3               ; 3 Zellen für Datenpuffer
TOR:   DB   OCH                ; Tabelle der peripheren LED-
                               ; Adressen
       DB   ODH
       DB   OEH
       DB   OFH
       DB   1CH
       DB   1DH
STAB:  DB   5FH                ; Segmenttabelle
       DB   6H
       DB   3BH
       DB   2EH
       DB   66H
       DB   6DH
       DB   7DH
       DB   7H
       DB   7FH
       DB   6FH
       DB   77H
       DB   7CH
       DB   59H
       DB   3EH
       DB   79H
       DB   71H
ZKO:   DB   60H                ; Zeitkonstante für CTC-Kanal 0
ZK1:   DB   64H                ; Zeitkonstante für CTC-Kanal 1
       END
```

3.5. Rechenprogramme

Die folgenden ausgewählten Zahlenprogramme sollen zeigen, wie man die Bit-Muster, aus denen die Befehle des Prozessors gebildet sind, für Zahlenrechnungen verwenden kann.

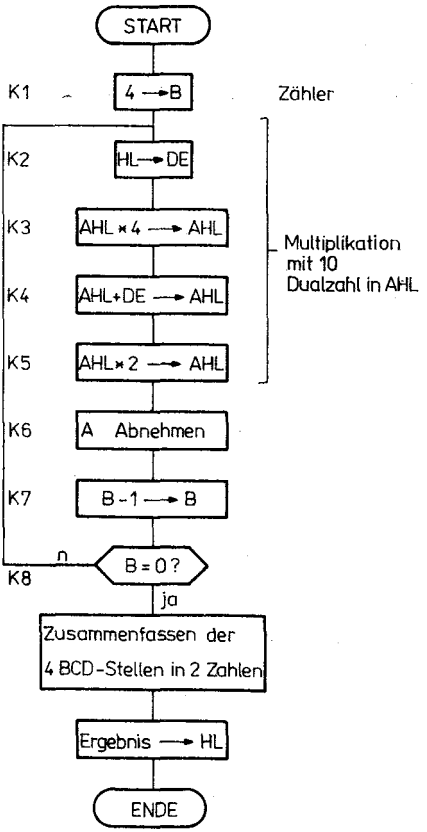


Bild 3.8
Flußdiagramm zur Umwandlung einer 16-Bit-Dualzahl in eine BCD-Zahl mit 4 Stellen

3.5.1. Umwandlung einer Dualzahl in eine BCD-Zahl

Multipliziert man eine Zahl $|z| < 1$ mit 10, so tritt die erste Dezimalstelle nach dem Komma vor das Komma. Diese Stelle wird abgenommen und anschließend durch weitere Multiplikation mit 10 die nächste Dezimalstelle ermittelt.
Die Dualzahl stehe im Register HL. Für die Dezimalziffer wird Register A als Zwischenspeicher benutzt. Die 4 Ergebnisziffern werden erst im Speicher zwischengespeichert und am Ende wieder nach HL gebracht.
Soll eine ganze Dualzahl in den BCD-Code mit 4 Dezimalstellen umgewandelt werden, so kann man die Dualzahl vorher durch 10 000 dividieren und dann das gleiche Verfahren anwenden.
Bild 3.8 zeigt das Flußdiagramm.

```
                                PN      B1
                        ; Programm  Umwandlung  DUAL-BCD
                        ; Dualzahl  Z = 0, Z1 ... Z16 in HL
                        ; BCD Zahl D = 0, D1D2D3D4 in HL
0000    DD214200 DUD1:  LD      IX, ZW                          ; Zwischenspeicher-
                                                                  adresse nach IX
0004    0604            LD      B, 4                            ; für 4 Dezimalstellen
0006    AF              XOR     A
0007    5D       K2:    LD      E, L                            ; Multiplikation mit 10
0008    54              LD      D, H
0009    CB15            RL      L                               ; 4 AHL = Multi-
                                                                  plikation mit 4
000B    CB14            RL      H
000D    CB17            RL      A
000F    CB15            RL      L
0011    CB14            RL      H
0013    CB17            RL      A
0015    19              ADD     HL, DE                          ; Dualzahl zuaddieren=
                                                        ; Multiplikation mit 5
0016    CE00            ADC     0
0018    CB15            RL      L                               ; 2 AHL = Multiplikation
                                                        ; mit 2-Gesamtmultiplikation MIT10
001A    CB14            RL      H
001C    CB17            RL      A
001E    DD7700          LD      ⟨IX,⟩ A                         ; Ziffer nach Zwischen-
                                                                  speicher
0021    DD23            INC     IX
0023    AF              XOR     A
0024    05              DEC     B
0025    20E0            JRNZ    K2-#
0027    214400          LD      HL, ZW + 2                      ; Adresse für 3. BCD-
                                                                  Stelle nach HL
```

```
0029  002A  3A4500          LD    A, ⟨ZW + 3⟩          ; 4. BCD-Stelle nach A
0030  002D  ED6F            RLD                        ; 3. und 4. BCD-Stelle
                                                        nach ZW + 2
0031  002F  23              INC   HL                   ; 1. BCD-Stelle nach A
0032  0030  3A4200          LD    A, ⟨ZW⟩
0033  0033  ED6F            RLD                        ; 1. Stelle nach ZW + 3
0034  0035  3A4300          LD    A, ⟨ZW + 1⟩          ; 2. Stelle nach A
0035  0038  ED6F            RLD                        ; 1. und 2. BCD-Stelle
                                                        nach ZW + 3
0036  003A  2A4400          LD    HL, ⟨ZW + 2⟩   ; Ergebnis nach HL
0037  003D  09              RET
0038  003E        ZW:       BER   4
0039                        END
Keine Syntax-Fehler CRAS 4200—K1520
```

3.5.2. Umwandlung einer BCD-Zahl in eine Dualzahl

Fall 1: Es liegt eine positive ganze 4stellige Dezimalzahl vor. Die Zahl habe die Form $d_3\ d_2\ d_1\ d_0$, wobei d_3, d_2, d_1, d_0 die BCD-Stellen sind.

Dann gilt

$z = d_3 \cdot 10^3 + d_2 \cdot 10^2 + d_1 \cdot 10 + d_0$,

$z = [(d_3 \cdot 10^2 + d_2) \cdot 10 + d_1] \cdot 10 + d_0$,

d. h., man muß d_3 abtrennen, mit 10 multiplizieren, d_2 addieren, wieder mit 10 multiplizieren usw.

Fall 2: Es liegt eine 4stellige echt gebrochene Dezimalzahl vor. Die Zahl hat die Form $z = 0, D_1\ D_2\ D_3\ D_4$, wobei D_1, D_2, D_3, D_4 die BCD-Stellen sind.

Die Umrechnung ins Dualsystem erfolgt nach dem Prinzip der Multiplikation mit 2 und Abtrennen der vor das Komma fallenden Dualstelle.

Bild 3.9 zeigt das Flußdiagramm für Fall 1 und Bild 3.10 für Fall 2.

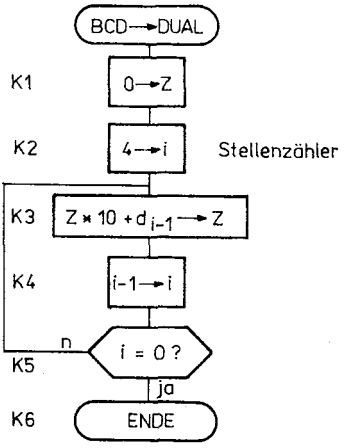


Bild 3.9
Flußdiagramm zur Umwandlung einer 4stelligen ganzen BCD-Zahl in eine Dualzahl

Programmliste BCD-DUAL Verfahren 1

```
001                 /Umwandlung BCD DUAL für ganze
                    /Zahlen
002                 /Die BCD-Zahl steht in HL – Die
                    /Dualzahl am Ende in HL
```

```
003  000 000 042 16 DED: LD (ZW+2),   /BCD-Zahl nach
     000 001 075          HL          /ZW+2 und ZW+3
     000 002 000
004  000 003 041 10       LD HL, ZW+2 /Adresse der BCD-
     000 004 075                      /Zahl nach HL
     000 005 000
005  000 006 257 04       XOR A
006  000 007 355 18       RLD         /Abtrennen der 1.
                                      /BCD-Ziffer
     000 010 157
007  000 011 062 13       LD (ZW), A
     000 012 073
     000 013 000
008  000 014 355 18       RLD         /Abtrennen der
     000 015 157                      /2. BCD-Ziffer
009  000 016 062 13       LD (ZW+1), A
     000 017 074
     000 020 000
010  000 021 043 06       INC HL      /Abtrennen der 3.
                                      /BCD-Ziffer
011  000 022 355 18       RLD
     000 023 157
012  000 024 062 13       LD (ZW+2), A
     000 025 075
     000 026 000
013  000 027 355 18       RLD         /Abtrennen der
     000 030 157                      /4. BCD-Ziffer
014  000 031 062 13       LD (ZW+3), A
     000 032 076
     000 033 000
015  000 034 335 14       LD IX, ZW   /Adresse Zwischen-
     000 035 041                      /speicher nach IX
     000 036 073
     000 037 000
016  000 040 041 10       LD HL, 0    /Z=0 Ausgangs-
     000 041 000                      /wert für die
     000 042 000                      /Dualzahl
017  000 043 006 07       LD B, 4     /Für 4 BCD-
     000 044 004                      Ziffern
018  000 045 135 04 K3:   LD E, L     Z × 10
```

```
019  000 046 124 D4      LD D, H
020  000 047 313 08      RL L
     000 050 025
021  000 051 313 08      RL H
     000 052 024
022  000 053 313 08      RL L
     000 054 025
023  000 055 313 08      RL H
     000 056 024
024  000 057 031 11      ADD HL, DE
025  000 060 313 08      RL L            /Multiplikation mit 2
     000 061 025
026  000 062 313 08      RL H
     000 063 024
027  000 064 335         LD E, (IX)      /BCD-Ziffer nach E
     000 065 136
     000 066 000
028  000 067 026         LD D, 0
     000 070 000
029  000 071 031 11      ADD HL, DE      /Addiere BCD-Ziffer
030  000 072 335         INC IX
     000 073 043
031  000 074 005 04      DEC B
032  000 075 040 12      JRNZ K3- #
     000 076 346
033  000 077 311 10      RET
034  000 100        ZW:  BER 4
```

Programmliste BCD-DUAL Verfahren 2

```
001                      /UMWANDLUNG BCD→DUAL
002                      /BCD-ZAHL STEHT ALS 0,
                                    D1D2D3D4 IN HL
003                      /DUAL-ZAHL KOMMT NACH DE
004  000 000 006 07 BD:  LD B, 10H       /Zähler für 16
     000 001 020                         /Dualstellen
005  000 002 175 04 K2:  LD A, L         /BCD-Zahl* 2
006  000 003 205 04      ADD L
007  000 004 047 04      DAA
008  000 005 157 04      LD L, A
009  000 006 174 04      LD A, H
```

```
010  000 007 214 04      ADC H
011  000 010 047 04      DAA
012  000 011 147 04      LD H, A
013  000 012 313 08      RL E          /Eintragen der
     000 013 023                       /Dualstellen in DE
014  000 014 313 08      RL D
     000 015 022
015  000 016 005 04      DEC B
016  000 017 040 12      JRNZ K2- #
     000 020 361
017  000 021 311 10      RET
GESAMTTAKTANZAHL = 000 081
```

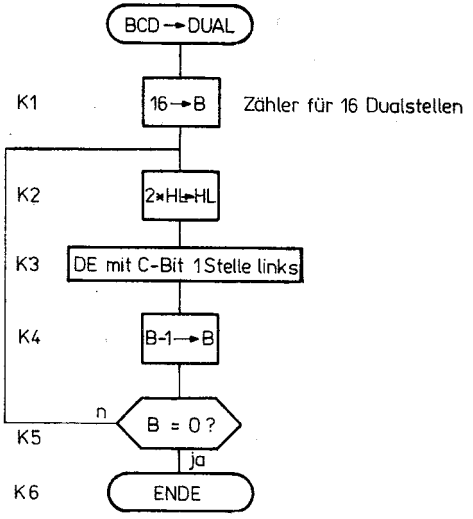


Bild 3.10 Flußdiagramm zur Umwandlung einer 4stelligen BCD-Zahl $|z| < 1$ in eine 16stellige Dualzahl

3.5.3. Programm zur Multiplikation von 2 positiven 16-Bit-Zahlen

Durch das folgende Programm werden 2 Dualzahlen mit einer Länge von maximal 16 Bit multipliziert. Der Multiplikator muß

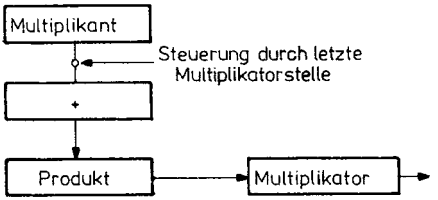


Bild 3.11 Funktionsprinzip für die Multiplikation

in dem Gemeinschaftsregister BC stehen; der Multiplikand in dem Registerpaar DE. Das Produkt ist 32 Bit lang und steht in den Registern HLBC. Bild 3.11 zeigt das Prinzipschema.

Programmbeschreibung
Zur Multiplikation wird der Multiplikator eine Stelle nach rechts geschoben. Dabei gelangt das niederwertigste Bit in das C-Flag. Ist dieses Bit 1, so wird der Inhalt von DE zu HL addiert, ist es 0, so erfolgt keine Addition von DE zu HL. Anschließend wird der Inhalt von HL mit BC gemeinsam um eine Stelle nach rechts verschoben, dabei kommt das nächste Bit des Multiplikators ins C-Flag. Dieser Vorgang wird insgesamt 16mal wiederholt, da der Multiplikator 16 Stellen umfaßt.
Bild 3.12 zeigt das Flußdiagramm für das Multiplikationsprogramm. Das Programm selbst hat folgendes Aussehen:

```
001                          /MULTIPLIKATION VON ZWEI
                             /16-BIT-ZAHLEN
002                          /MULTIPLIKATOR IN BC
003                          /MULTIPLIKAND DE
004                          /PRODUKT IN HLBC
005  000 000 041 10 MUL2: LD HL, 0
     000 001 000
     000 002 000
006  000 003 076 07      LD A, 11H      /Zähler für 16
     000 004 021                        /Stellen
007  000 005 313 08 K3: RR B            /BC nach rechts
     000 006 030                        /verschieben
008  000 007 313 08      RR C
     000 010 031
```

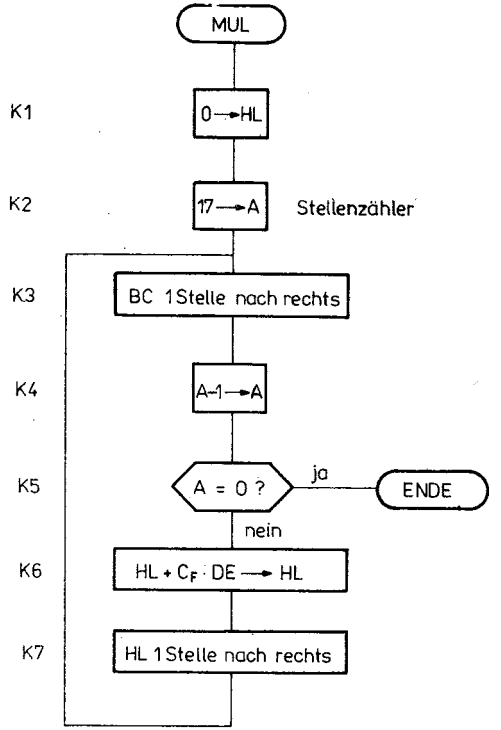


Bild 3.12 Flußdiagramm für die Multiplikation von 2 16stelligen Dualzahlen

```
009   000 011 075 04        DEC A
010   000 012 310 11        RZ
011   000 013 060 12        JRNC K7- #
      000 014 001
012   000 015 031 11        ADD HL, DE
013   000 016 313 08 K7:    RR H            /HL nach rechts
      000 017 034                           /verschieben
014   000 020 313 08        RR L
      000 021 035
```

```
015  000 022 030 12      JR K3- #
     000 023 361
GESAMTTAKTANZAHL = 000 099
```

3.5.4. Division

Das folgende Programm bringt die Division von 2 Zahlen A:B = Q, wobei A und B 15 Dualstellen lang sein können. Es muß jedoch A < B und A < 1 und B < 1 sein.

Das Verfahren ist aus dem Prinzipschema (Bild 3.13) zu erschen. Vor Beginn wird das Quotientenregister gelöscht.

1. Bildung von Dividend — Divisor
   Ist das Ergebnis negativ, so wird der Divisor wieder zum Ergebnis dazugezählt (Rückstellung des Restes) und in die letzte Stelle des Quotienten eine 0 eingetragen.
   Ist das Ergebnis positiv, so wird keine Rückstellung des Restes vorgenommen. In die letzte Quotientenstelle trägt man eine 1 ein.
2. Quotient und Dividend werden gemeinsam 1 Stelle nach links verschoben.
3. 1. und 2. werden so oft wiederholt, wie der Quotient Stellen haben soll.

Am Ende steht der Quotient im Quotientenregister und im Dividendenregister der Rest.

Im konkreten Fall soll für den Divisor das Registerpaar DE, für den Dividend das Registerpaar HL und für den Quotienten das Registerpaar BC verwendet werden.

Bild 3.14 zeigt das Flußdiagramm der Division.

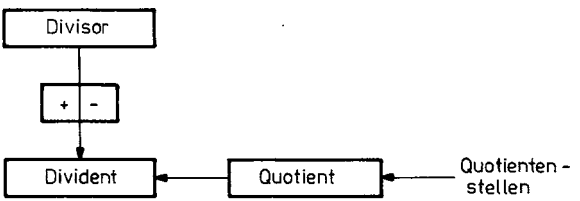


Bild 3.13 Funktionsprinzip für die Division

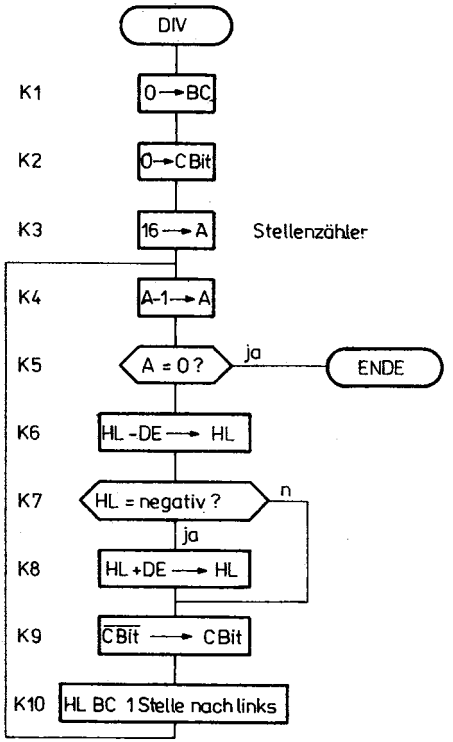


Bild 3.14 Flußdiagramm für die Division von 2 15stelligen Dualzahlen (bei K3 lies 17→A)

```
0001                            PN      D1
0002                    ; Division
0003                    ; Dividend 15 Stellen nach HL
0004                    ; Divisor 15 Stellen nach DE
0005  0000  010000  DI2R  LD      BC, 0
0006  0003  AF            XOR     A               ; C-FLAG löschen
0007  0004  3E11          LD      A, 11H          ; für 15 Stellen
0008  0006  3D      K4:   DEC     A
0009  0007  2810          JRZ     K11-#
0010  0009  ED52          SBC     HL, DE
0011  000B  3001          JRNC    K9-#
0012  000D  19            ADD     HL, DE          ; Rückstellung
0013  000E  3F      K9:   CCF                     ; Quotientenstelle
0014  000F  CB11          RL      C               ; Ins C-FLAG
0015  0011  CB10          RL      B
0016  0013  CB15          RL      L
0017  0015  CB14          RL      H
0018  0017  18ED          JR      K4-#
0019  0019  C9      K11:  RET
0020                    END
```

Kein Syntax-Fehler CRAS 4200-K1520

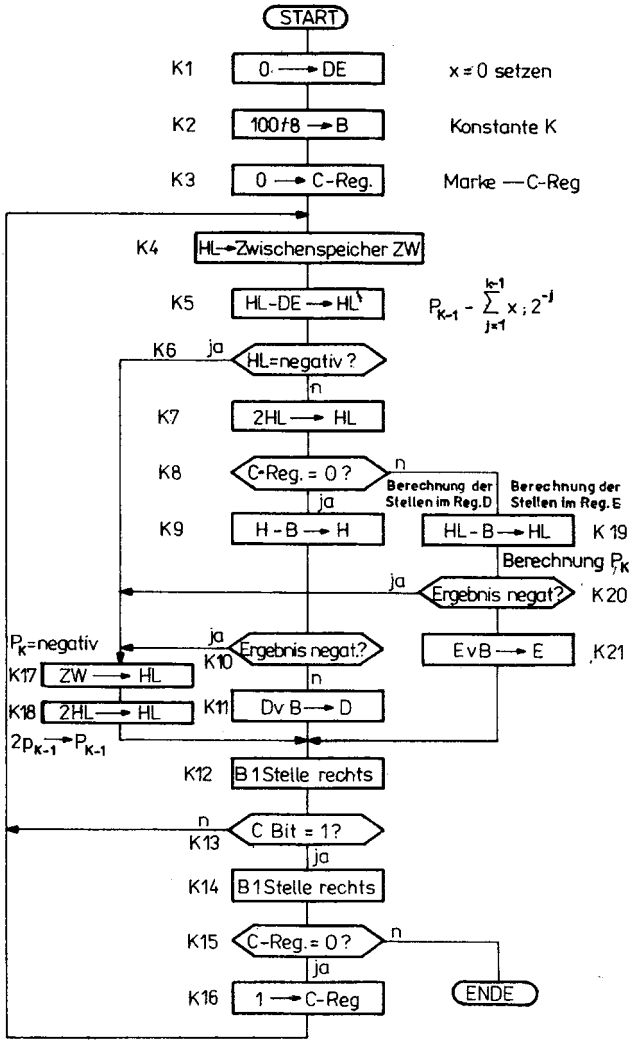


Bild 3.15 Flußdiagramm für das Quadratwurzelziehen aus einer 15stelligen Dualzahl mit $|z| < 1$

3.5.5. Quadratwurzel

Das folgende Programm soll zeigen, wie mathematische Aufgaben auf einem Mikrorechner programmiert werden können. Wichtig dabei ist, daß man zunächst ein Lösungsverfahren findet, das möglichst an das Dualsystem angepaßt ist. Es gilt:

$Z = X$

$Z = 0, Z_1Z_2Z_3 \ldots Z_N \qquad X = 0, X_1X_2X_3 \ldots X_N$

Verfahren

$P_0 = Z \qquad X_0 = 0$

$$P_K = 2\left[P_{K-1} - \operatorname{sgn}(P_{K-1})\,(2^{-K} + \sum_{j=1}^{K-1} X_j \cdot 2^{-J}) + 2^{-K-1}\right] . \quad (1)$$

$$X_K = \frac{1 + \operatorname{sgn}(P_K)}{2} \qquad \operatorname{sgn}(P) = \begin{cases} +1, \text{ falls P positiv} \\ -1, \text{ falls P negativ} \end{cases}$$

Das Verfahren ist ein Iterationsverfahren. Die Zahl Z, aus der die Wurzel gezogen wird, muß kleiner als 1 sein. Sie liegt als reine Dualzahl mit n Dualstellen Z_1, Z_2 bis Z_n vor. Die Wurzel aus Z hat wieder n Dualstellen X_1, X_2 bis X_N.

Im Verfahren geht man von der Zahl $P_0 = Z$ aus. Die Ziffer X_0 (Stelle vor dem Komma) ist 0. Mit P_0 und X_0 wird nach den angegebenen Formeln P_1 und X_1 berechnet. X_1 ist die 1. Stelle der Lösung hinter dem Komma. Mit P_1 und X_1 berechnet man P_2 und X_2 usw. Mit jedem Schritt gewinnt man eine Dualstelle der Wurzel.

Um das Verfahren „rechnergerecht" aufzubereiten, wird die Formel etwas umgestellt. Es sei angenommen, daß P_0 P_1 bis P_{K-1} und X_0 X_1 bis X_{K-1} bereits ermittelt worden sind und nun P_K und X_K ermittelt werden sollen.

Wenn man P_{K-1} als Zahl schreibt, gilt:

$$P_{K-1} = 0, S_1S_2 \ldots S_{K-1}\, S_K\, S_{K+1} \ldots S_n , \quad (2)$$

wobei $S_1S_2 \ldots S_n$ die Dualstellen von P_{K-1} sind.

Weiterhin gilt:

$$\sum_{j=1}^{K-1} X_j \cdot 2^{-1} = 0, X_1 X_2 \ldots X_{K-1} . \quad (3)$$

Ist P_{K-1} positiv, so ist $\operatorname{sgn}(P_{K-1}) = 1$ und

$$P_K = 2\left[P_{K-1} - \sum_{j=1}^{K-1} X_j\, 2^{-j} - 2^{-K-1}\right], \qquad P_K = 2\left[P_{K-1} - \sum_{j=1}^{K-1} X_j\, 2^{-1}\right] - 2^{-K}\,. \tag{4}$$

Ist P_{K-1} negativ, so ist $\operatorname{sgn}(P_{K-1}) = -1$ und

$$P_K = 2\left[P_{K-1} + \sum_{j=1}^{K-1} X_j\, 2^{-j} + 2^{-K} + 2^{-K-1}\right]. \tag{5}$$

Wird P_K nach Gl. (4) negativ, so wird jedoch P_{K+1} nicht nach Gl. (5) berechnet, sondern es gilt mit P_K negativ
$\operatorname{sgn}(P_K) = -1;\ X_K = 0$:

$$P_{K+1} = 2\left[2P_K + \sum_{j=1}^{K} X_j\, 2^{-j} + 2^{-K-1} + 2^{-K-2}\right], \text{ nach (5)}$$

$$P_{K+1} = 2\left[P_{K-1} - \sum_{j=1}^{K-1} X_j\, 2^{-j} - 2^{-K-1}\right]. \text{ nach (4)}$$

$$P_{K+1} = 2\left[2P_{K-1} - 2\sum_{j=1}^{K-1} X_j\, 2^{-j} - 2^{-K} + \underbrace{\sum_{j=1}^{K-1} X_j\, 2^{-j}}_{\text{da } X_K = 0} + 2^{-K-1} + 2^{-K+2}\right],$$

$$P_{K+1} = 2\left[2\,P_{K-1} - \sum_{j=1}^{K-1} X_j\, 2^{-j} - 2^{-K-2}\right],$$

$$P_{K+1} = 2\left[2\,P_{K-1} - \sum_{j=1}^{K-1} X_j\, 2^{-j}\right] - 2^{-K-1}. \tag{6}$$

Mit Gl. (4) und Gl. (6) erhält man folgendes Verfahren zum Radizieren von Quadratwurzeln:

		Dualwert
1.	Voraussetzungen	
a)	Bereitstellen von $P_0 = Z$	$0\ Z_1\ Z_2 \ldots Z_N$
b)	$X = 0,\ X_1X_2 \ldots X_n = 0$ setzen	$0\ 0\ 0 \ldots 0$
c)	Bereitstellen einer Konstanten K zum Bit der Ziffer X_K	$0\ 1\ 0\ 0 \ldots 0$
2.	Rechenablauf	
	Ausgangspunkt P_{K-1}; am Anfang P_0	
a)	Bildung von $2P_{K-1}$ für Gl. (6)	

b) Bildung von $P_K = 2\left[P_{K-1} - \sum_{j=1}^{K-1} X_j 2^{-j}\right] - 2^{-K}$

c) Ist P_K positiv, so wird $X_K = 1$ gesetzt (Bildung von $X \dotplus K \rightarrow X$), und statt P_{K-1} wird P_K gesetzt.

d) Ist P_K negativ, so wird $X_K = 0$ gesetzt, und statt P_{K-1} wird für Gl. (6) $2P_{K-1}$ gesetzt [für Gl. (6)].

e) K wird um eine Stelle nach rechts verschoben.

f) Fortführung bei a) so lange, bis die Anzahl der Stellen genügt.

Das folgende Beispiel zeigt das Programm für die Quadratwurzel aus einer 15stelligen Dualzahl $|z| < 1$.

```
0001                                ; Wurzelprogramm
0002
0003                                ; Radikand 0, D1 ... D15   In HL
0004                                ; Wurzel    0, W1 ... W15   In DE
0005
0006                                PN      W2
0007  0000  010040    WU2R:         LD      BC, 4000H
0008  0003  110000                  LD      DE, 0
0009  0006  224200    K4:           LD      ⟨ZW⟩, HL
0010  0009  AF                      XOR     A
0011  000A  ED52                    SBC     HL, DE
0012  000C  381D                    JRC     K17-#                   ; Abfrage K6
0013  000E  CB15                    RL      L
0014  0010  CB14                    RL      H
0015  0012  AF                      XOR     A
0016  0013  A9                      XOR     C
0017  0014  201E                    JRNZ    K19-#                   ; Abfrage K8
0018  0016  7C                      LD      A, H
0019  0017  90                      SUB     B
0020  0018  67                      LD      H, A
0021  0019  3810                    JRC     K17-#                   ; Abfrage K10
0022  001B  7A                      LD      AD
0023  001C  B0                      OR      B
0024  001D  57                      LD      D, A
0025  001E  CB18    K12:           RR      B
0026  0020  30E4                    JRNC    K4-#                    ; Abfrage K13
0027  0022  CB18                    RR      B
0028  0024  AF                      XOR     A
0029  0025  81                      ADD     C
0030  0026  CO                      RNZ                             ; Abfrage K15
0031  0027  0E01                    LD      C, 1
0032  0029  18DB                    JR      K4-#
```

```
0033   002B   2A4200   K17:   LD     HL,⟨ZW⟩
0034   002E   CB25            SLA    L
0035   0030   CB14            RL     H
0036   0032   18EA            JR     K12-#
0037   0034   7D       K19;   LD     A,L
0038   0035   90              SUB    B
0039   0036   6F              LD     L,A
0040   0037   7C              LD     A,H
0041   0038   DE00            SBC    0
0042   003A   67              LD     H,A
0043   003B   38EE            JRC    K17-#          ; Abfrage K20
0044   003D   7B              LD     A,E
0045   003E   B0              OR     B
0046   003F   5F              LD     E,A
0047   0040   18DC            JR     K12-#
0048   0042   0000    ZW:     DA     0
0049                          END
Keine Syntax-Fehler CRAS 4200—K1520
```

4. Schlußbetrachtung

In den drei Heften ist auf eine Reihe von Grundlagen der Mikrorechentechnik eingegangen worden. Dabei sind diese Grundlagen vorwiegend auf das Bausteinsystem der *U 880*-Reihe, die in der DDR am meisten verbreitet ist, bezogen. Die Schaltkreise eines Mikrorechners werden in der Zukunft noch einen großen Aufschwung erhalten. Der Grund dafür ist die hohe Flexibilität, mit der sich Mikrorechnerkonfigurationen an alle denkbaren Probleme anpassen lassen können. Es gibt heute in der Welt schon eine große Anzahl solcher Reihen von Schaltkreisen für Mikrorechner, die alle aus den Schaltkreisen Prozessor, paralleler und serieller Ein-/Ausgabe-Baustein, DMA-Baustein und Zeitgeberbaustein bestehen. Solche Systeme sind z. B. *Motorola M 6800, INTEL 8085, Valvo 2640, MC 9900*. Bei weiterentwickelten Systemen ist der Speicher mit in den Prozessor integriert. Es entstehen dann sogenannte Mikrocomputerschaltkreise. Der Mikrocomputerschaltkreis *Z 8* von *Zilog* besteht z. B. aus Mikroprozessor, 2K Byte ROM, 0,25 K Byte RAM, Ein-/Ausgabe-Toren, Taktgenerator und Zeitsteuerung. Die Wortlänge tendiert in der weiteren Entwicklung neben 8 Bit auf 16 und 32 Bit.

Durch die rasche Entwicklung der Halbleitertechnologie zu immer höheren Integrationsgraden entstehen auch Schaltkreise für komplizierte Rechnerfunktionen, z. B. Gleitkommaprozessoren, Schaltkreise für DA- und AD-Wandlung und Schaltkreise für mathematische Standardfunktionen wie Radizieren, Sinus, Cosinus, Tangens, Logarithmus und Potenzieren. Beispielsweise gibt es zum Aufbau von Tischrechnern vollständige Arithmetikprozessoren. In vielen Tischrechnern werden sogenannte CAP-Chips (Cordic-Arithmetic-Prozessor-Chip) verwendet.

Der Einsatz solcher Rechenschaltkreise erfordert ein grundsätzliches Umdenken in der Digitalelektronik. Funktionen werden nicht mehr verdrahtet, sondern programmiert. Es wird ein Grundsystem (eines Mikrorechners) aus Rechenschaltkreisen aufgebaut, und dieses System, das sehr universell einsetzbar ist, programmiert man dann auf den Einsatzfall. Der Aufbau des Mikrorechnersystems geschieht dabei nach durch die Schaltkreisfa-

milien vorgegebenen Regeln. Lediglich vom Umfang des Systems her ergeben sich bestimmte Unterschiede. Man kann Mikrorechnersysteme entsprechend der Größe des Einsatzfalles einteilen in:

a) Einkartenrechner,
b) Mikrorechnersysteme,
c) Mehrprozessorsysteme,
d) Hierarchische Rechnersysteme mit verteilter Intelligenz.

Einkartenrechner

Die einfachste Variante eines Mikrorechners besteht in der Zusammenschaltung eines Mikroprozessors mit Speicherbausteinen und mit Ein- und Ausgabebausteinen zur Verbindung mit exter-

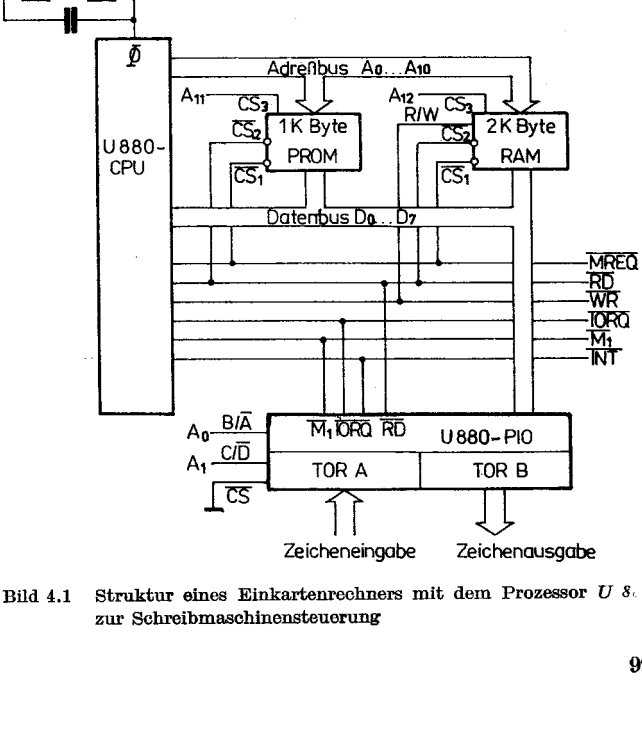


Bild 4.1 Struktur eines Einkartenrechners mit dem Prozessor *U 880* zur Schreibmaschinensteuerung

nen Geräten. Die dazugehörenden Schaltkreise lassen sich bei modernen mikroelektronischen Bausteinen unmittelbar zusammenschalten und auf einer Leiterkarte unterbringen. Man erhält auf diese Weise den Einkartenrechner, der als Steuerrechner für viele Anwendungsfälle ausreicht.
Bild 4.1 zeigt die prinzipielle Schaltung eines solchen Einkartenrechners mit dem Prozessor *U 880*, dem PIO-Baustein und mit 2 K PROM und 1 K RAM.
Der Taktgenerator für diesen Rechner braucht bei Frequenzen unter 1 MHz nicht stabilisiert zu werden. Der angegebene Leiterkartenrechner läßt sich sehr vielseitig anwenden.
Zum Beispiel ist es möglich, ihn zur Steuerung einer elektrischen Schreibmaschine, deren Typenhebel durch Magnete ausgelöst werden, zu verwenden. Man kann auf diese Weise eine Schreibmaschine entwickeln, die den Text von einer Seite speichert. Durch eine geeignete Programmierung erreicht man, daß in dem gespeicherten Text beliebig korrigiert werden kann. Am Schluß einer Korrektur läßt sich der Inhalt so ausschreiben, wie er benötigt wird. Auf ganz ähnliche Weise kann man einen solchen Einkartenrechner zur Steuerung von Zeichengeräten, in Oszillografen sowie in Datenerfassungsgeräten einsetzen.

Mikrorechnersystem

Das Mikrorechnersystem ist ein abgestimmtes Sortiment von Leiterkarten mit unterschiedlichen Funktionen, die an einen zentralen Bus, den sogenannten Systembus, angeschlossen sind (Bild 4.2). Der Systembus unterteilt sich in
Adreßbus,
Datenbus,
Steuerbus.
Der Adreßbus enthält eine Speicheradresse oder die Adresse eines Ein- und Ausgabetores. Über den Datenbus wird der Datenaustausch zwischen den einzelnen Moduln (Leiterkarten) vollzogen. Der Datenbus ist bidirectional (arbeitet in beiden Richtungen!). Der Steuerbus enthält eine Reihe von Steuersignalen zur Bedienung der adressierten Module sowie Meldesignale über den momentanen Zustand (z. B. bereit, nicht bereit, fehlerhaft) des adressierten Moduls.

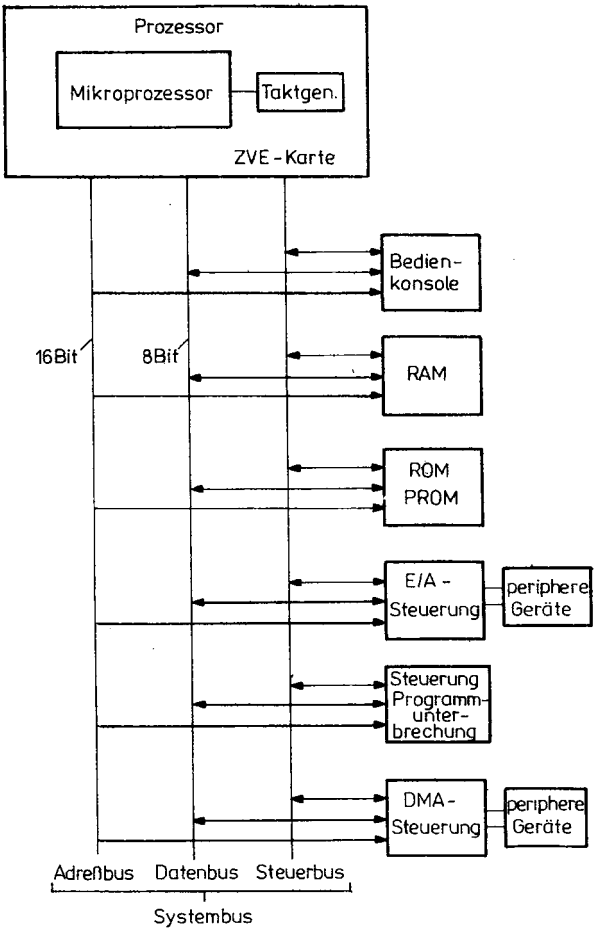
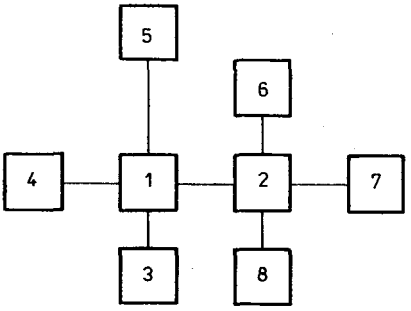


Bild 4.2 Aufbau eines Mikrorechnersystems

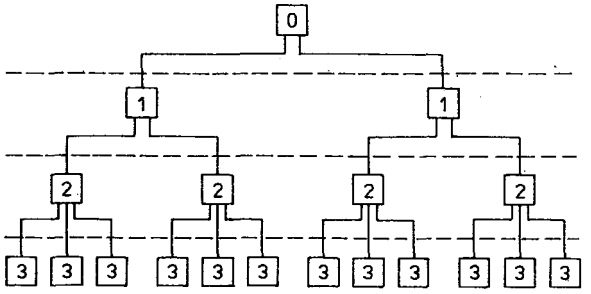
Mehrprozessorsysteme

Da im Mikroprozessor der vollständige Rechnerkern als Chip zur Verfügung gestellt wird, können bei relativ geringen Kosten in einem System mehrere Prozessoren verwendet werden.



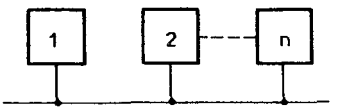
Sternstruktur

a)

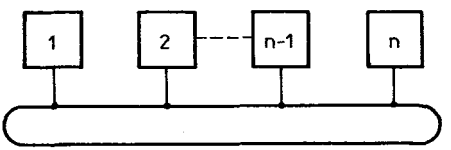


Hierarchiestruktur

b)

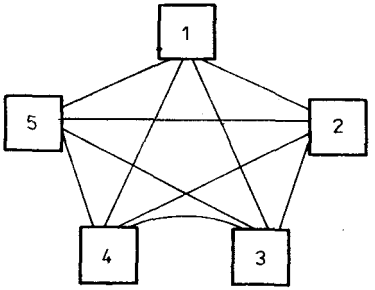


Busstruktur



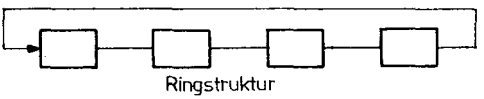
Ringbus

c)



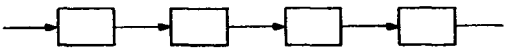
Netzstruktur

d)



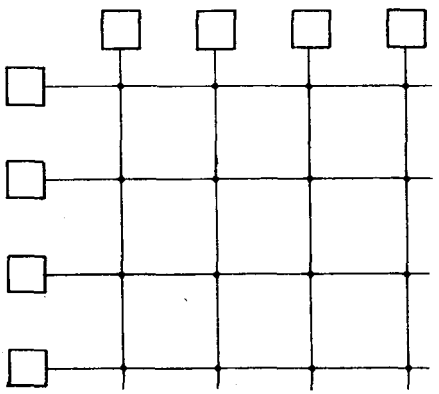
Ringstruktur

e)



Reihenstruktur

f)



Matrixstruktur

g)

Bild 4.3 Schaltungsvarianten für Mehrprozessorsysteme; a — Sternstruktur, b — Hierarchiestruktur, c — Busstruktur, d — Netzstruktur, e — Ringstruktur, f — Reihenstruktur, g — Matrixstruktur

Die Verwendung von mehreren Prozessoren hat aber nur dann einen Sinn, wenn von der Aufgabenstellung her eine entsprechende Aufteilung in Teilaufgaben möglich ist. Die Aufteilung in Teilaufgaben kann dabei z. B. so aussehen, daß eine größere Aufgabe in Teilaufgaben zerlegt wird und jeder Mikroprozessor praktisch eine solche Aufgabe abarbeitet.

Man kommt durch diese Aufteilung zu sogenannten Hierarchischen Systemen. Zur Kommunikation der einzelnen Prozessoren miteinander (Austausch von Parametern, Aufgabenteilung und Fertigmeldung) müssen diese entsprechend verbunden werden. Dafür gibt es eine ganze Reihe von Zusammenschaltungsprinzipien.

Bild 4.3 zeigt einige Möglichkeiten der Zusammenschaltung von Prozessoren. Bei der sternförmigen Zusammenschaltung (Bild 4.3a) läuft der Informationsaustausch über den Sternpunkt. Jeder Prozessor ist am Sternpunkt angeschlossen. Er kann seine Information über diesen an andere Prozessoren weitergeben. In Bild 4.3b ist eine Hierarchische Struktur dargestellt. In dieser gibt es zu einer Reihe von Prozessoren einen übergeordneten Prozessor, über den die entsprechenden Aufgaben verteilt und die Ergebnisse in Empfang genommen werden. Die Hierarchische Struktur kann auch als Mehrebenenstruktur aufgebaut werden. Bei der Busstruktur (Bild 4.3c) sind die einzelnen Prozessoren an einen zentralen BUS angeschlossen. Damit ein Prozessor mit einem 2. Prozessor in Verbindung treten kann, muß vom 1. Prozessor eine Busanforderung gestellt werden. Nachdem die Busanforderung angenommen wurde, wird der aufrufende Prozessor zum Master und die Gegenstelle zum Slave erklärt. Der Bus ist nun für den Datenaustausch frei.

Das System läßt sich so aufbauen, daß alle Elemente gleichberechtigt sind oder daß für die Busanforderung ein Prioritätssystem existiert. Der Nachteil dieser Schaltung besteht darin, daß immer nur 2 Elemente gleichzeitig miteinander arbeiten können.

Das Bussystem ist z. B. in der SKR-Reihe (System von Kleinrechnern) verbreitet.

Bei der Netzstruktur (Bild 4.3d) ist jedes Prozessorelement mit jedem verbunden. Die Struktur gestattet einen flexiblen Einsatz, jedoch ist sie sehr aufwendig in den Verbindungsstrecken. In Bild 4.3e ist die Ringstruktur dargestellt. Bei dieser Struktur

werden die Information und die Zieladresse über den Ringbus gegeben. Das Prozeßelement mit der gewünschten Adresse übernimmt die über den Bus laufenden Informationen. Dieses System ist für den Aufbau eines Netzes, bei dem die einzelnen Prozessorelemente örtlich weiter voneinander aufgestellt sein können, sehr zweckmäßig, z. B. die komplexe Steuerung von Maschinen in Werkhallen oder Industrieeinrichtungen. Das System wird im international bekannten CAMAC-System angewendet.
Die Reihenstruktur (Bild 4.3f) ähnelt der Ringstruktur, jedoch ist die Kette nicht geschlossen.
Aus Bild 4.3g ist die Matrixstruktur zu ersehen. Bei ihr läßt sich jedes Prozeßelement mit jedem anderen über die Matrixknotenpunkte verbinden. Die Knotenpunkte sind als elektronische Schalter ausgeführt, die von einem Steuerrechner des Netzwerks durchgeschaltet werden.
Eine weitere Anwendung der Mehrprozessorsysteme ist die Parallelverarbeitung. Dabei werden mehrere Prozessoren so in Kette zusammengefügt, daß sie alle den gleichen Anteil zur Realisierung eines Vorgangs leisten. Als Beispiel stelle man sich eine Stahlplatte vor, die an einer Seite (x-Richtung) ungleichmäßig erwärmt wird. Das System soll nun ermitteln, wie sich die Erwärmung in y-Richtung fortpflanzt. Zu diesem Zweck wird die x-Richtung in einzelne Teilabschnitte aufgeteilt (x_1 bis x_n). Jeden Teilabschnitt stellt man durch ein Prozessorelement dar. Das Programm der einzelnen Elemente basiert in jedem Prozessor auf den Regeln der Wärmeleitung, so daß die Verteilung in y-Richtung als Zeitkonstante bei der Programmbearbeitung dargestellt wird.

Hierarchische Rechnersysteme (Rechnernetze)

Bei den Rechnernetzen werden Mikrorechnersysteme miteinander gekoppelt. Die Kopplung dient dem Zweck der besseren Verteilung der Aufgaben zur gleichmäßigen Auslastung der einzelnen Rechner. Das wichtigste Element eines solchen Rechnernetzes ist das dazugehörige Betriebssystem, das die Aufgaben und die Geräte verwaltet und steuert. Auf diese Art und Weise können große Aufgaben mit umfangreichem Datenmaterial gelöst werden.
Bild 4.4 zeigt die Struktur eines solchen Rechnernetzes. Zum

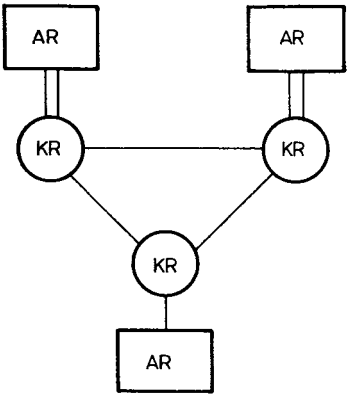


Bild 4.4 Struktur eines Rechnernetzes

Netz gehören die Arbeitsrechnersysteme und ein sogenannter Kommunikationsrechner zur Steuerung der Verbindungen.
Der häufigste Anwendungsfall der Rechenschaltkreise wird der Einkartenrechner sein. Er dient als eingebauter Steuerrechner für einzelne Geräte. Das Mikrorechnersystem wird als Prozeßrechner bei größeren Steuerungsproblemen angewendet werden.
Mehrprozessorsysteme werden seltener auftreten. Sie sind dort von Nutzen, wenn es um die Nachbildung von physikalischen Modellen geht. Hierarchische Rechnersysteme finden in größeren Systemen Anwendung, z. B. als Verbundnetz bei einer Flugplatzüberwachung.
Bei der Frage nach der möglichen Anwendung von Mikroprozessoren kann man heute schon antworten: Sie sind überall einsetzbar. Wichtig für ein erfolgreiches Anwenden ist die Auswahl des geeigneten Prozessors und des geeigneten Systems sowie der verfügbaren Programme. Die vorliegenden Hefte sollen dazu eine gewisse Hilfestellung leisten.

Höhne, M.: Der Mikroprozessor U 808 D; radio-fernsehen-elektronik, 26 (1977) H. 5

RFT-Information Zentrale Verarbeitungseinheit U 808 D, Firmendokumentation; Kombinat VEB Funkwerk Erfurt, 1977

RFT-Information Zentrale Verarbeitungseinheit U 880 D, Firmendokumentation; Kombinat VEB Funkwerk Erfurt

Mikroprozessoren, Mikrorechner; Literaturinformation des VEB Kombinat Robotron (ZfT), Dresden '77

Z 80 — Entwicklungssystem; Firmenunterlagen Mostek, Wien 1977

Mikroprozessor-Schaltkreise 8080; Firmendokumentation Siemens, München 1977

Mikrorechnersystem K 1510 — Systembeschreibung; Firmendokumentation 1977, VEB Kombinat Robotron, ZfT, Dresden

Betriebsdokumentation des Mikrorechnersystems K 1510; Dokumentation des VEB Kombinat Robotron Dresden, 1977

Mikrorechnersystem K 1520 — Systembeschreibung;

Firmendokumentation 1978, VEB Kombinat Robotron Dresden

Mikroprozessoren; Elektronik-Sonderausgabe, Franzis-Verlag, München 1977

Intel MCS Microcomputer Set; Users Manual, Fa. Intel (USA) 1973

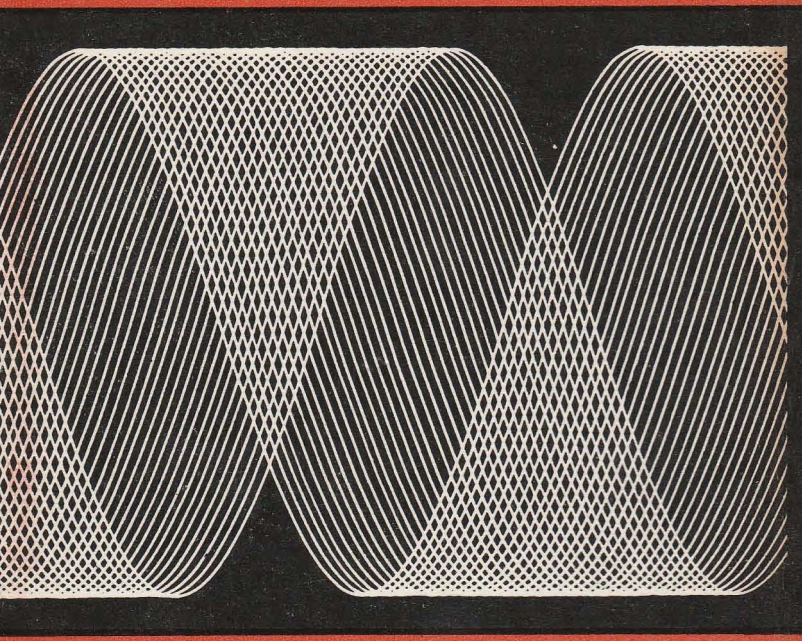
Kretzschmar, J. G.: Gestaltung des Ein-Ausgabesystems von Mikrorechnern; Sozialistische Rationalisierung, 6 (1977) H. 7

Werner, D.: Besonderheiten der Programmierung von Mikrorechnern; Nachrichtentechnik-Elektronik, 27 (1977) H. 2

Koch, G. R.: Stand und Trends der Programmierung von Mikroprozessoren, Elektronik 1977, H. 1

Jossifow, W.: Programmierung für Mikroprozessoren; Dipl.-Arb. TH Ilmenau, Sektion TBK, 1977

Liebich, W.: Mikroprozessoren und Mikrorechner; in Philippow: Taschenbuch der Elektrotechnik in 6 Bänden, Bd. 4; VEB Verlag Technik, Berlin 1978

Abraham, W., Pfeiffer, V.: Rechnerstrukturen zur Realisierung

digitaler Steuerungen; Z. elektrische Inform. — Energietechnik 7 (1977), S. 400 bis 403
Dawidczak, S., Weise, K. D.: Mikrorechnersystem robotron K 1510; messen-steuern-regeln, 20 (1977), H. 12
Friebel, W., Roloff, H., Schiller, H., Vogt, H.: Programmierung von Mikroprozessoren; Reihe Automatisierungstechnik 182, VEB Verlag Technik, Berlin 1978
Katzarow, S.: Mikroprozessorgesteuerter Lehrautomat auf der Grundlage elektronischer Module; Dissertation TH Ilmenau, Sektion TBK, 1977
Zaremba, J.: Mikroprozessoren und Mikrorechner; Fernmeldetechnik, 18 (1978), H. 1 ff.
Meiling, W.: Einige Tendenzen in der Entwicklung und Anwendung von Mikroprozessoren und Mikrorechnern; Nachrichtentechnik, 27 (1977), H. 1
Mierau, W., Lindner, S.: Technik der Mikrorechner; radio-fernsehen-elektronik, 26 (1977), H. 17 ff.
Werner, D.: Programmieren von Mikrorechnern; Rechentechnik, Datenverarbeitung 6 (1976)
Reiher, Th.: Ein Mikroprogrammaufbereitungssystem auf der Basis des U 808; Sozialistische Rationalisierung, 6 (1977), H. 7
Keller, D., Weise, K. D.: Mikroprozessoren; Wiss. Inf. und Berichte Nr. 5, 1977, VEB Kombinat Robotron (ZfT) Dresden
Kanton, D.: Mikroprozessorsysteme, Reihe Automatisierungstechnik 183, VEB Verlag Technik, Berlin 1978
Applikation Mikrorechner; Kombinat VEB Funkwerk Erfurt, 1977
Reichel, R.: Mikroprozessoren: radio-fernsehen-elektronik, H. 1 (1977), S. 5 bis 7
Gößler, R., Schwerte, J.; Auf dem Weg zur Mikroprozessor-Praxis; Elektronik (1976), H. 3, S. 74 bis 86